

IBM[®] DB2[®] Universal Database



Consulta de SQL

Versión 7

IBM[®] DB2[®] Universal Database



Consulta de SQL

Versión 7

Antes de utilizar esta información y el producto al que da soporte, asegúrese de leer la información general incluida en el "Apéndice S. Avisos" en la página 1649.

Esta publicación es la traducción del original inglés *IBM DB2 SQL Reference Version 7 (SC09-2974-01)*.

Este documento contiene información sobre productos patentados de IBM. Se proporciona de acuerdo con un contrato de licencia y está protegido por la ley de la propiedad intelectual. La presente publicación no incluye garantías del producto y las declaraciones que contiene no deben interpretarse como tales.

Puede solicitar publicaciones a través del representante de IBM o sucursal de IBM de su localidad, o bien llamando a los números de teléfono 1-800-879-2755, en los Estados Unidos, o 1-800-IBM-4YOU, en Canadá.

Cuando envía información a IBM, otorga a IBM un derecho no exclusivo para utilizar o distribuir dicha información en la forma en que IBM considere adecuada, sin contraer por ello ninguna obligación con el remitente.

© Copyright International Business Machines Corporation 1993, 2001. Reservados todos los derechos.

Contenido

Capítulo 1. Introducción	1	Interfaz de nivel de llamada (CLI) de DB2 y	
Quién debe utilizar este manual.	1	Open Database Connectivity (ODBC)	31
Forma de utilizar este manual	1	Programas JDBC (Java Database Connectivity)	
Estructura de este manual.	1	y SQLJ (SQL incorporado para Java)	32
Lectura de los diagramas de sintaxis	3	Paquetes	32
Convenios utilizados en este manual	6	Vistas de catálogo	32
Condiciones de error	6	Conversión de caracteres.	33
Convenios de resaltado	6	Juegos de caracteres y páginas de códigos	34
Documentación relacionada para este manual	6	Atributos de páginas de códigos	35
		Supervisores de sucesos	37
Capítulo 2. Conceptos	9	Desencadenantes	37
Definiciones de base de datos relacional	9	Espacios de tablas y otras estructuras de	
Lenguaje de consulta estructurada (SQL)	9	almacenamiento.	39
Autorización y privilegios	10	Partición de datos entre múltiples particiones	40
Esquemas	13	Mapas de particionamiento	42
Control de la utilización de esquemas	13	Colocación de tablas	43
Tablas	14	Base de datos relacional distribuida	43
Vistas	15	Servidores de aplicaciones	44
Seudónimos	16	CONNECT (Tipo 1) y CONNECT (Tipo 2)	45
Índices	16	Unidad de trabajo remota	45
Claves	17	Unidad de trabajo distribuida dirigida por	
Claves de unicidad.	17	aplicación.	49
Claves primarias	17	Consideraciones acerca de la	
Claves foráneas	17	representación de datos	55
Claves de particionamiento	17	Sistemas federados DB2	56
Restricciones	18	Servidor federado, base de datos federada	
Restricciones de unicidad	18	y fuentes de datos	56
Restricciones de referencia	19	Tareas realizadas en un sistema federado	
Restricciones de comprobación de tabla	22	DB2.	56
Nivel de aislamiento	23	Reiniciadores y módulos de reiniciador	59
Lectura repetible (RR).	24	Definiciones de servidor y opciones de	
Estabilidad de lectura (RS)	25	servidor	59
Estabilidad del cursor (CS)	26	Correlaciones de usuarios y opciones de	
Lectura no confirmada (UR).	26	usuario.	62
Comparación de los niveles de aislamiento	26	Correlaciones de tipos de datos	62
Consultas	26	Correlaciones de funciones, plantillas de	
Expresiones de tabla	27	funciones y opciones de correlación de	
Expresiones de tabla comunes	27	funciones	63
Procesos, simultaneidad y recuperación de		Apodos y opciones de columna	64
aplicaciones	27	Especificaciones de índice	65
SQL interactivo	30	Peticiónes distribuidas	66
SQL incorporado	30	Compensación	67
SQL estático	30	Paso a través.	68
SQL dinámico	31		
		Capítulo 3. Elementos del lenguaje	69

Caracteres	69	TIMESTAMP	123
Consideraciones acerca de MBCS	70	DATALINK	123
Símbolos	70	Tipos definidos por el usuario	123
Consideraciones acerca de MBCS	71	Atributo de posibilidad de nulo del	
Identificadores	71	resultado.	124
Identificadores SQL	72	Reglas para las conversiones de series	124
Identificadores de sistema principal	72	Compatibilidad entre particiones.	127
Convenios de denominación y calificaciones		Constantes	129
implícitas de nombres de objetos	72	Constantes enteras	129
Seudónimos	78	Constantes de coma flotante	129
ID de autorización y nombres-autorización.	79	Constantes decimales	130
Características del SQL dinámico en la		Constantes de tipo serie.	130
ejecución	81	Constantes hexadecimales	130
Los ID de autorización y la preparación de		Constantes gráficas de tipo serie	131
sentencias	82	Utilización de constantes con tipos	
Tipos de datos	82	definidos por el usuario.	131
Nulos	84	Registros especiales	132
Gran objeto (LOB)	84	CLIENT ACCTNG	132
Series de caracteres	87	CLIENT APPLNAME	132
Series gráficas	88	CLIENT USERID	133
Serie binaria	89	CLIENT WRKSTNNAME	133
Números	90	CURRENT DATE	134
Valores de indicación de fecha y hora	91	CURRENT DEFAULT TRANSFORM	
Valores DATALINK	95	GROUP	134
Tipos definidos por el usuario	96	CURRENT DEGREE	135
Promoción de los tipos de datos	100	CURRENT EXPLAIN MODE	136
Conversión entre tipos de datos	101	CURRENT EXPLAIN SNAPSHOT	137
Asignaciones y comparaciones	105	CURRENT NODE	138
Asignaciones numéricas.	106	CURRENT PATH	138
Asignaciones de series	107	CURRENT QUERY OPTIMIZATION	139
Asignaciones de indicación de fecha y		CURRENT REFRESH AGE.	140
hora	110	CURRENT SCHEMA	140
Asignaciones de DATALINK	111	CURRENT SERVER	141
Asignaciones de los tipos definidos por el		CURRENT TIME	141
usuario	113	CURRENT TIMESTAMP	142
Asignaciones de tipos de referencia	114	CURRENT TIMEZONE	142
Comparaciones numéricas	114	USER	143
Comparaciones de series	115	Nombres de columna	143
Comparaciones de indicación de fecha y		Nombres de columna calificados.	144
hora	119	Nombres de correlación.	144
Comparaciones de tipos definidos por el		Calificadores de nombres de columna	
usuario	119	para evitar ambigüedad.	147
Comparaciones de tipos de referencia	120	Calificadores de nombres de columna en	
Reglas para los tipos de datos del resultado	120	referencias correlacionadas.	149
Series de caracteres	121	Referencias a variables del lenguaje principal	151
Series gráficas	122	Variables del lenguaje principal en el SQL	
Gran objeto binario (BLOB)	122	dinámico.	152
Numérico	122	Referencias a las variables del lenguaje	
DATE	123	principal BLOB, CLOB y DBCLOB	154
TIME	123	Referencias a variables localizadoras	155

Referencias a las variables de referencia a archivos BLOB, CLOB y DBCLOB	155	Predicado IN	220
Referencias a variables de lenguaje principal de tipo estructurado.	158	Predicado LIKE	223
Funciones	159	Predicado NULL	229
Funciones definidas por el usuario externas, de SQL y derivadas	160	Predicado TYPE	230
Funciones definidas por el usuario: escalares, de columna, de fila y de tabla	160	Condiciones de búsqueda	232
Signaturas de función	161	Ejemplos.	234
Vía de acceso de SQL	161	Capítulo 4. Funciones	235
Resolución de funciones	162	Funciones de columna	257
Invocación de función	167	AVG	258
Métodos	168	CORRELATION	260
Métodos definidos por el usuario: externos y SQL	168	COUNT	261
Signaturas de método	169	COUNT_BIG	263
Invocación de métodos	169	COVARIANCE.	265
Resolución de métodos	170	GROUPING	266
Método de elección de la mejor opción	171	MAX	268
Ejemplo de resolución de método	173	MIN	270
Invocación de métodos	173	Funciones REGRESSION	272
Semántica de enlace conservador	174	STDDEV	276
Expresiones.	176	SUM	277
Sin operadores.	177	VARIANCE	278
Con el operador de concatenación	177	Funciones escalares	279
Con operadores aritméticos	180	ABS o ABSVAL	280
Dos operandos enteros	182	ACOS.	281
Operandos enteros y decimales	182	ASCII	282
Dos operandos decimales	182	ASIN	283
Aritmética decimal en SQL.	182	ATAN.	284
Operandos de coma flotante	183	ATAN2	285
Tipos definidos por el usuario como operandos	183	BIGINT	286
Selección completa escalar	184	BLOB	287
Operaciones de fecha y hora y duraciones	184	CEILING o CEIL	288
Aritmética de fecha y hora en SQL	185	CHAR	289
Prioridad de operaciones	190	CHR	295
Expresiones CASE	190	CLOB.	296
Especificaciones CAST	193	COALESCE.	297
Operaciones de deferencia	196	CONCAT	298
Funciones OLAP	197	COS	299
Invocación de métodos	204	COT	300
Tratamiento de los subtipos	206	DATE.	301
Referencia de secuencia	207	DAY	303
Predicados	212	DAYNAME	305
Predicado básico	213	DAYOFWEEK	306
Predicado cuantificado	214	DAYOFWEEK_ISO	307
Predicado BETWEEN	217	DAYOFYEAR	308
Predicado EXISTS.	219	DAYS	309
		DBCLOB.	310
		DECIMAL	311
		DECRYPT_BIN y DECRYPT_CHAR	314
		DEGREES	316
		DEREF	317

DIFFERENCE	318	NODENUMBER	390
DIGITS	319	NULLIF	392
DLCOMMENT	320	PARTITION	393
DLINKTYPE	321	POSSTR	395
DLURLCOMPLETE	322	POWER	398
DLURLPATH	323	QUARTER	399
DLURLPATHONLY	324	RADIANS	400
DLURLSCHEME	325	RAISE_ERROR	401
DLURLSERVER	326	RAND	403
DLVALUE	327	REAL	404
DOUBLE	329	REC2XML	405
ENCRYPT	331	REPEAT	411
EVENT_MON_STATE	334	REPLACE	412
EXP	335	RIGHT	413
FLOAT	336	ROUND	414
FLOOR	337	RTRIM	416
GETHINT	338	RTRIM (esquema SYSFUN)	417
GENERATE_UNIQUE	339	SECOND	418
GRAPHIC	341	SIGN	419
HEX	342	SIN	420
HOUR	344	SMALLINT	421
IDENTITY_VAL_LOCAL	345	SOUNDEX	422
INSERT	351	SPACE	423
INTEGER	353	SQRT	424
JULIAN_DAY	355	SUBSTR	425
LCASE o LOWER	356	TABLE_NAME	429
LCASE (esquema SYSFUN)	357	TABLE_SCHEMA	431
LEFT	358	TAN	434
LENGTH	359	TIME	435
LN	361	TIMESTAMP	436
LOCATE	362	TIMESTAMP_ISO	438
LOG	363	TIMESTAMPDIFF	439
LOG10	364	TRANSLATE	441
LONG_VARCHAR	365	TRUNCATE o TRUNC	444
LONG_VARGRAPHIC	366	TYPE_ID	445
LTRIM	367	TYPE_NAME	446
LTRIM (esquema SYSFUN)	368	TYPE_SCHEMA	447
MICROSECOND	369	UCASE o UPPER	448
MIDNIGHT_SECONDS	370	VALUE	449
MINUTE	371	VARCHAR	450
MOD	372	VARGRAPHIC	452
MONTH	373	WEEK	454
MONTHNAME	374	WEEK_ISO	455
MQPUBLISH	375	YEAR	456
MQREAD	378	Funciones de tabla	457
MQRECEIVE	380	MQREADALL	458
MQSEND	382	MQRECEIVEALL	461
MQSUBSCRIBE	384	SQLCACHE_SNAPSHOT	464
MQUNSUBSCRIBE	386	Procedimientos	465
MULTIPLY_ALT	388	GET_ROUTINE_SAR	466

PUT_ROUTINE_SAR	467	ALTER USER MAPPING	610
Funciones definidas por el usuario	469	ALTER VIEW	613
Capítulo 5. Consultas	471	BEGIN DECLARE SECTION	615
subselección	472	CALL	617
cláusula-select	473	CLOSE	626
cláusula-from	478	COMMENT	628
referencia-tabla.	479	COMMIT	640
tabla-unida	483	SQL compuesto (dinámico)	642
cláusula-where.	486	SQL compuesto (incorporado).	648
Cláusula group-by	487	CONNECT (Tipo 1)	653
cláusula-having	494	CONNECT (Tipo 2)	662
Ejemplos de subselecciones	496	CREATE ALIAS	670
Ejemplos de uniones	499	CREATE BUFFERPOOL.	673
Ejemplos de conjuntos de agrupación, Cube		CREATE DISTINCT TYPE	677
y Rollup	503	CREATE EVENT MONITOR	684
selección completa	512	CREATE FUNCTION	695
Ejemplos de una selección completa	515	CREATE FUNCTION (Escalar externa).	696
sentencia-select	518	CREATE FUNCTION (Tabla externa)	723
expresión-común-tabla	519	CREATE FUNCTION (Tabla externa OLE	
cláusula-order-by	522	DB)	741
cláusula-update	526	CREATE FUNCTION (fuente o plantilla)	750
cláusula-read-only	527	CREATE FUNCTION (SQL, escalar, de tabla	
cláusula-fetch-first	528	o de fila).	761
cláusula-optimize-for.	529	CREATE FUNCTION MAPPING	769
Ejemplos de una sentencia-select.	530	CREATE INDEX	774
Capítulo 6. Sentencias de SQL.	533	CREATE INDEX EXTENSION	782
Cómo se invocan las sentencias de SQL	537	CREATE METHOD	790
Incorporación de una sentencia en un		CREATE NICKNAME	795
programa de aplicación	538	CREATE NODEGROUP.	801
Preparación y ejecución dinámicas	539	CREATE PROCEDURE	804
Invocación estática de una sentencia-select	540	CREATE SCHEMA	822
Invocación dinámica de una		CREATE SEQUENCE	826
sentencia-select	540	CREATE SERVER	831
Invocación interactiva	541	CREATE TABLE	835
Utilización de SQL con otros sistemas		CREATE TABLESPACE	893
principales	541	CREATE TRANSFORM.	903
Códigos de retorno de SQL	542	CREATE TRIGGER	910
SQLCODE	542	CREATE TYPE (Estructurado).	923
SQLSTATE	543	CREATE TYPE MAPPING	950
comentarios de SQL	544	CREATE USER MAPPING.	955
ALTER BUFFERPOOL	545	CREATE VIEW	957
ALTER NICKNAME	548	CREATE WRAPPER	974
ALTER NODEGROUP	552	DECLARE CURSOR	976
ALTER SEQUENCE	556	DECLARE GLOBAL TEMPORARY TABLE	982
ALTER SERVER	561	DELETE	991
ALTER TABLE.	565	DESCRIBE	997
ALTER TABLESPACE	597	DISCONNECT	1002
ALTER TYPE (Estructurado)	603	DROP	1005
		END DECLARE SECTION	1034
		EXECUTE	1036

EXECUTE IMMEDIATE	1042
EXPLAIN	1045
FETCH	1050
FLUSH EVENT MONITOR	1054
FREE LOCATOR.	1055
GRANT (autorizaciones de base de datos)	1056
GRANT (privilegios de índice)	1059
GRANT (privilegios de paquete)	1061
GRANT (privilegios de esquema)	1064
GRANT (privilegios de secuencia)	1067
GRANT (Privilegios de servidor)	1068
GRANT (privilegios de apodo, vista o tabla)	1070
GRANT (privilegios para espacios de tablas)	1078
INCLUDE	1080
INSERT.	1082
LOCK TABLE.	1092
OPEN	1094
PREPARE	1099
REFRESH TABLE	1110
RELEASE (Conexión)	1111
RELEASE SAVEPOINT.	1113
RENAME TABLE	1114
RENAME TABLESPACE	1116
REVOKE (autorizaciones de bases de datos)	1118
REVOKE (privilegios de índice).	1122
REVOKE (privilegios de paquete)	1124
REVOKE (privilegios de esquema).	1127
REVOKE (Privilegios de servidor)	1129
REVOKE (privilegios de apodo, vista o tabla)	1131
REVOKE (privilegios para espacios de tablas)	1137
ROLLBACK	1139
SAVEPOINT	1142
SELECT.	1144
SELECT INTO	1145
SET CONNECTION	1147
SET CURRENT DEFAULT TRANSFORM GROUP.	1149
SET CURRENT DEGREE	1151
SET CURRENT EXPLAIN MODE	1153
SET CURRENT EXPLAIN SNAPSHOT	1155
SET CURRENT PACKAGESET	1157
SET CURRENT QUERY OPTIMIZATION	1159
SET CURRENT REFRESH AGE.	1162
SET ENCRYPTION PASSWORD	1164
SET EVENT MONITOR STATE	1166
SET INTEGRITY.	1168

SET PASSTHRU	1179
SET PATH	1181
SET SCHEMA	1184
SET SERVER OPTION	1186
SET variable	1188
UPDATE	1193
VALUES	1204
VALUES INTO	1205
WHENEVER	1207

Capítulo 7. Sentencias de control de SQL 1209

Sentencia de procedimiento SQL	1210
Sentencia ALLOCATE CURSOR	1212
Sentencia de asignación	1214
Sentencia ASSOCIATE LOCATORS	1216
Sentencia CASE	1218
Sentencia FOR	1221
Sentencia GET DIAGNOSTICS	1223
Sentencia GOTO.	1225
Sentencia IF	1227
Sentencia ITERATE	1229
Sentencia LEAVE	1230
Sentencia LOOP	1231
Sentencia compuesta (procedimiento).	1233
Sentencia REPEAT	1240
Sentencia RESIGNAL	1242
Sentencia RETURN	1245
Sentencia SIGNAL	1248
Sentencia WHILE	1251

Apéndice A. Límites de SQL 1253

Apéndice B. Comunicaciones SQL (SQLCA) 1261

Visualización de SQLCA interactivamente	1261
Descripciones de los campos de la SQLCA	1261
Orden del informe de errores	1265
Utilización de la SQLCA por DB2 Enterprise - Extended Edition	1266

Apéndice C. Área de descriptores SQL (SQLDA) 1267

Descripciones de los campos.	1268
Campos en la cabecera SQLDA	1269
Campos de una ocurrencia de una SQLVAR base	1270
Campos de una ocurrencia de una SQLVAR secundaria	1272
Efecto de DESCRIBE en la SQLDA.	1274

SQLTYPE y SQLLEN	1276
SQLTYPES no soportados y no reconocibles	1278
Números decimales empaquetados	1279
Campo SQLLEN para decimal	1280
Apéndice D. Vistas de catálogo	1281
Vistas de catálogo actualizables	1282
‘Guía’ de las vistas de catálogo	1282
‘Guía’ de las vistas de catálogo actualizables	1285
SYIBM.SYSDUMMY1	1286
SYSCAT.ATTRIBUTES	1287
SYSCAT.BUFFERPOOLNODES	1289
SYSCAT.BUFFERPOOLS	1290
SYSCAT.CASTFUNCTIONS	1291
SYSCAT.CHECKS	1292
SYSCAT.COLAUTH	1293
SYSCAT.COLCHECKS	1294
SYSCAT.COLDIST	1295
SYSCAT.COLOPTIONS	1296
SYSCAT.COLUMNS	1297
SYSCAT.CONSTDEP	1303
SYSCAT.DATATYPES	1304
SYSCAT.DBAUTH	1306
SYSCAT.EVENTMONITORS	1308
SYSCAT.EVENTS	1310
SYSCAT.FULLHIERARCHIES	1311
SYSCAT.FUNCDEP	1312
SYSCAT.FUNCMAPOPTIONS	1313
SYSCAT.FUNCMAPPARMOPTIONS	1314
SYSCAT.FUNCMAAPPINGS	1315
SYSCAT.FUNCPARMS	1316
SYSCAT.FUNCTIONS	1318
SYSCAT.HIERARCHIES	1324
SYSCAT.INDEXAUTH	1325
SYSCAT.INDEXCOLUSE	1326
SYSCAT.INDEXDEP	1327
SYSCAT.INDEXES	1328
SYSCAT.INDEXOPTIONS	1332
SYSCAT.KEYCOLUSE	1333
SYSCAT.NAMEMAPPINGS	1334
SYSCAT.NODEGROUPDEF	1335
SYSCAT.NODEGROUPS	1336
SYSCAT.PACKAGEAUTH	1337
SYSCAT.PACKAGEDEP	1338
SYSCAT.PACKAGES	1339
SYSCAT.PARTITIONMAPS	1344
SYSCAT.PASSTHROUGHAUTH	1345
SYSCAT.PROCEDURES	1346

SYSCAT.PROCOPTIONS	1349
SYSCAT.PROCPARMOPTIONS	1350
SYSCAT.PROCPARMS	1351
SYSCAT.REFERENCES	1353
SYSCAT.REVTYPEMAPPINGS	1354
SYSCAT.SCHEMAAUTH	1356
SYSCAT.SCHEMATA	1357
SYSCAT.SEQUENCES	1358
SYSCAT.SERVEROPTIONS	1360
SYSCAT.SERVERS	1361
SYSCAT.STATEMENTS	1362
SYSCAT.TABAUTH	1363
SYSCAT.TABCONST	1365
SYSCAT.TABLES	1366
SYSCAT.TABLESPACES	1371
SYSCAT.TABOPTIONS	1373
SYSCAT.TBSPACEAUTH	1374
SYSCAT.TRIGDEP	1375
SYSCAT.TRIGGERS	1376
SYSCAT.TYPEMAPPINGS	1378
SYSCAT.USEROPTIONS	1380
SYSCAT.VIEWDEP	1381
SYSCAT.VIEWS	1382
SYSCAT.WRAPOPTIONS	1383
SYSCAT.WRAPPERS	1384
SYSSTAT.COLDIST	1385
SYSSTAT.COLUMNS	1386
SYSSTAT.FUNCTIONS	1388
SYSSTAT.INDEXES	1390
SYSSTAT.TABLES	1394

Apéndice E. Vistas de catálogo para utilizar con tipos estructurados 1397

‘Guía’ de las vistas de catálogo	1399
OBJCAT.INDEXES	1400
OBJCAT.INDEXEXPLOITRULES	1404
OBJCAT.INDEXEXTENSIONDEP	1405
OBJCAT.INDEXEXTENSIONMETHODS	1406
OBJCAT.INDEXEXTENSIONPARMS	1407
OBJCAT.INDEXEXTENSIONS	1408
OBJCAT.PREDICATESPECS	1409
OBJCAT.TRANSFORMS	1410

Apéndice F. Sistemas federados 1411

Tipos de servidor	1411
Opciones SQL para sistemas federados	1412
Opciones de columna	1413
Opciones de correlación de funciones	1414
Opciones de servidor	1415
Opciones de usuario	1420

Correlaciones de tipos de datos por omisión	1421
Correlaciones de tipos por omisión entre fuentes de datos DB2 y DB2 Universal Database para OS/390 (y DB2 para MVS/ESA)	1422
Correlaciones de tipos por omisión entre fuentes de datos DB2 y 2 Universal Database para AS/400 (y DB2 para OS/400)	1422
Correlaciones de tipos por omisión entre fuentes de datos DB2 y Oracle	1422
Correlaciones de tipos por omisión entre fuentes de datos DB2 y DB2 para VM y VSE (y SQL/DS)	1423
Proceso del recurso de paso a través	1423
Proceso SQL en sesiones de paso a través	1423
Consideraciones y restricciones	1424

Apéndice G. Tablas de la base de datos de ejemplo 1427

La base de datos de muestra.	1428
Para crear la base de datos SAMPLE	1428
Para borrar la base de datos de muestra	1428
Tabla CL_SCHED	1429
Tabla DEPARTMENT	1429
Tabla EMPLOYEE	1429
Tabla EMP_ACT	1433
Tabla EMP_PHOTO	1435
Tabla EMP_RESUME	1435
Tabla IN_TRAY	1436
Tabla ORG	1436
Tabla PROJECT	1437
Tabla SALES	1438
Tabla STAFF	1439
Tabla STAFFG	1440

Archivos de muestra con tipos de datos BLOB y CLOB	1441
Foto de Quintana	1441
Currículum vitae de Quintana	1441
Foto de Nicholls	1442
Currículum vitae de Nicholls	1443
Foto de Adamson	1444
Currículum vitae de Adamson	1444
Foto de Walker	1445
Currículum vitae de Walker	1446

Apéndice H. Nombres de esquema reservados y palabras reservadas . . . 1447

Esquemas reservados 1447

Palabras reservadas	1447
Palabras reservadas de SQL de IBM	1449
Palabras reservadas de ISO/ANS SQL92	1451

Apéndice I. Comparación de niveles de aislamiento 1453

Apéndice J. Interacción de desencadenantes y restricciones . . . 1455

Apéndice K. Tablas de Explain y definiciones 1459

Tabla EXPLAIN_ARGUMENT	1460
Tabla EXPLAIN_INSTANCE	1464
Tabla EXPLAIN_OBJECT	1467
Tabla EXPLAIN_OPERATOR	1469
Tabla EXPLAIN_PREDICATE	1472
Tabla EXPLAIN_STATEMENT	1474
Tabla EXPLAIN_STREAM	1477
Tabla ADVISE_INDEX	1479
Tabla ADVISE_WORKLOAD	1482
Definiciones de tabla para tablas de Explain	1483
Definición de tabla EXPLAIN_ARGUMENT	1484
Definición de tabla EXPLAIN_INSTANCE	1485
Definición de tabla EXPLAIN_OBJECT	1486
Definición de tabla EXPLAIN_OPERATOR	1487
Definición de tabla EXPLAIN_PREDICATE	1488
Definición de tabla EXPLAIN_STATEMENT	1489
Definición de tabla EXPLAIN_STREAM	1490
Definición de tabla ADVISE_INDEX	1491
Definición de tabla ADVISE_WORKLOAD	1493

Apéndice L. Valores de registro de explicaciones 1495

Apéndice M. Ejemplo de recurrencia: Lista de material 1499

Ejemplo 1: Explosión de primer nivel	1499
Ejemplo 2: Explosión resumida	1501
Ejemplo 3: Control de profundidad	1502

Apéndice N. Tablas de excepciones 1505

Reglas para crear una tabla de excepciones 1505

Manejo de filas en las tablas de excepciones 1508

Consulta de las tablas de excepciones. . . . 1508

Apéndice O. Consideraciones acerca de EUC de japonés y de chino tradicional . 1511

Elementos del lenguaje.	1511
Caracteres	1511
Símbolos	1511
Identificadores	1512
Tipos de datos	1512
Asignaciones y comparaciones	1512
Reglas para los tipos de datos del resultado	1513
Reglas para las conversiones de series	1514
Constantes.	1515
Funciones	1515
Expresiones	1515
Predicados.	1516
Funciones	1517
LENGTH	1517
SUBSTR	1517
TRANSLATE	1517
VARGRAPHIC	1517
Sentencias	1518
CONNECT	1518
PREPARE	1518

Apéndice P. Especificaciones BNF para los enlaces de datos. 1521

Apéndice Q. Glosario 1525

Apéndice R. Utilización de la biblioteca de DB2 1629

Archivos PDF y manuales impresos sobre DB2	1629
Información sobre DB2	1629
Impresión de los manuales PDF	1640
Solicitud de los manuales impresos	1640
Documentación en línea de DB2	1642
Acceso a la ayuda en línea	1642
Visualización de información en línea	1644
Utilización de los asistentes de DB2	1646
Configuración de un servidor de documentos	1647
Búsqueda de información en línea	1648

Apéndice S. Avisos 1649

Marcas registradas	1652
------------------------------	------

Índice 1655

Cómo ponerse en contacto con IBM . 1689

Información sobre productos.	1689
--------------------------------------	------

Capítulo 1. Introducción

Este capítulo preliminar:

- Identifica la finalidad de este manual y las personas a las que va dirigido
- Explica el modo de utilizar el manual y su estructura
- Explica la notación de los diagramas de sintaxis, los convenios de denominación y de resaltado que se utilizan en el manual
- Lista la documentación relacionada
- Presenta una visión general de la familia de productos

Quién debe utilizar este manual

Este manual va dirigido a aquellas personas que deseen utilizar el Lenguaje de consulta estructurada (SQL) para acceder a una base de datos. Principalmente, es para los programadores y los administradores de bases de datos, pero también puede utilizarlo el usuario general que utiliza el procesador de línea de mandatos.

Este manual sirve más de consulta que de guía de aprendizaje. Supone que va a escribir programas de aplicación y, por lo tanto, presenta todas las funciones del gestor de bases de datos.

Forma de utilizar este manual

Este manual define el lenguaje SQL que utiliza DB2 Universal Database Versión 7. Utilícelo como un manual de consulta para informarse sobre los conceptos de bases de datos relacionales, elementos del lenguaje, las funciones, los formatos de las consultas y la sintaxis y semántica de las sentencias de SQL. Los apéndices se pueden utilizar para averiguar los límites e información sobre componentes importantes.

Estructura de este manual

Este manual de consulta está dividido en dos volúmenes. El Volumen 1 contiene las secciones siguientes:

- El “Capítulo 1. Introducción”, indica la finalidad, los destinatarios y la utilización del manual.
- El “Capítulo 2. Conceptos” en la página 9, explica los conceptos básicos de las bases de datos relacionales y del SQL.
- El “Capítulo 3. Elementos del lenguaje” en la página 69, describe la sintaxis básica del SQL y los elementos del lenguaje que son comunes a muchas sentencias de SQL.

- El “Capítulo 4. Funciones” en la página 235, contiene diagramas de sintaxis, descripciones semánticas, normas y ejemplos de utilización de las funciones de columna y funciones escalares del SQL.
- El “Capítulo 5. Consultas” en la página 471, describe los distintos formatos de una consulta.
- Los apéndices incluidos en el Volumen 1 contienen la información siguiente:
 - El “Apéndice A. Límites de SQL” en la página 1253, contiene las limitaciones del SQL.
 - El “Apéndice B. Comunicaciones SQL (SQLCA)” en la página 1261, contiene la estructura SQLCA.
 - El “Apéndice C. Área de descriptores SQL (SQLDA)” en la página 1267, contiene la estructura SQLDA.
 - El “Apéndice D. Vistas de catálogo” en la página 1281, contiene las vistas de catálogo de la base de datos.
 - El “Apéndice E. Vistas de catálogo para utilizar con tipos estructurados” en la página 1397, contiene las vistas de catálogo de tipo estructurado de la base de datos.
 - El “Apéndice F. Sistemas federados” en la página 1411, contiene opciones y correlaciones de tipos para sistemas federados.
 - El “Apéndice G. Tablas de la base de datos de ejemplo” en la página 1427, contiene las tablas de muestra utilizadas para los ejemplos.
 - El “Apéndice H. Nombres de esquema reservados y palabras reservadas” en la página 1447, contiene los nombres de esquemas reservados y las palabras reservadas correspondientes a los estándares SQL de IBM y SQL92 ISO/ANS.
 - El “Apéndice I. Comparación de niveles de aislamiento” en la página 1453, contiene un resumen de los niveles de aislamiento.
 - El “Apéndice J. Interacción de desencadenantes y restricciones” en la página 1455, explica la interacción de los desencadenantes y las restricciones referenciales.
 - El “Apéndice K. Tablas de Explain y definiciones” en la página 1459, contiene las tablas de Explain y cómo están definidas.
 - El “Apéndice L. Valores de registro de explicaciones” en la página 1495, describe la interacción entre sí de los valores de registro especiales CURRENT EXPLAIN MODE y CURRENT EXPLAIN SNAPSHOT y los mandatos PREP y BIND.
 - El “Apéndice M. Ejemplo de recurrencia: Lista de material” en la página 1499, contiene un ejemplo de una consulta recursiva.
 - El “Apéndice N. Tablas de excepciones” en la página 1505, contiene información sobre tablas creadas por el usuario que se utilizan con la sentencia SET INTEGRITY.

- El "Apéndice O. Consideraciones acerca de EUC de japonés y de chino tradicional" en la página 1511, lista las consideraciones cuando se utilizan los juegos de caracteres EUC.
- El "Apéndice P. Especificaciones BNF para los enlaces de datos" en la página 1521, contiene las especificaciones en formato BNF para valores DATALINK.

El Volumen 2 contiene las secciones siguientes:

- El "Capítulo 6. Sentencias de SQL" en la página 533, contiene diagramas de sintaxis, descripciones semánticas, normas y ejemplos de todas las sentencias de SQL.
- El "Capítulo 7. Sentencias de control de SQL" en la página 1209, contiene diagramas de sintaxis, descripciones semánticas, normas y ejemplos de sentencias de procedimiento de SQL.

Lectura de los diagramas de sintaxis

En este manual, la sintaxis se describe utilizando la estructura definida de la siguiente manera:

Lea los diagramas de sintaxis de izquierda a derecha y de arriba a abajo, siguiendo la línea.

El símbolo ►— indica el inicio de una sentencia.

El símbolo —► indica que la sintaxis de la sentencia continúa en la línea siguiente.

El símbolo ►— indica que continúa la sentencia de la línea anterior.

El símbolo —◄ indica el final de una sentencia.

Los elementos necesarios aparecen en la línea horizontal (la línea principal).

►—SENTENCIA—*elemento necesario*—◄

Los elementos opcionales aparecen debajo de la línea principal.

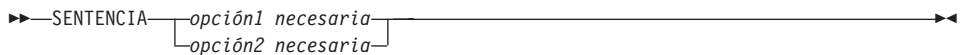
►—SENTENCIA—
└*elemento opcional*┘—◄

Si aparece un elemento opcional por encima de la línea principal, dicho elemento no tiene ningún efecto en la ejecución de la sentencia y sólo se utiliza para una lectura más sencilla.



Si puede elegir entre dos o más elementos, aparecen en una pila.

Si *debe* elegir uno de los elementos, un elemento de la pila aparece en la ruta principal.



Si se puede elegir no seleccionar ninguno de los elementos, toda la pila aparece por debajo de la línea principal.



Si uno de los elementos es el valor por omisión, aparecerá por encima de la línea principal y el resto de las opciones se mostrarán por debajo.



Una flecha que vuelve a la izquierda, por encima de la línea principal, indica un elemento que se puede repetir. En este caso, los elementos repetidos deben ir separados por uno o varios espacios en blanco.



Si la flecha de repetición contiene una coma, debe separar los elementos repetidos con una coma.



Una flecha de repetición por encima de una pila indica que puede elegir más de una opción de entre los elementos apilados o repetir una sola opción.

Las palabras clave aparecen en mayúsculas (por ejemplo, FROM). Deben escribirse exactamente igual a como aparecen en la sintaxis. Las variables aparecen en minúsculas (por ejemplo, nombre-columna). Representan nombres suministrados por el usuario o valores de la sintaxis.

Si aparecen signos de puntuación, paréntesis, operadores aritméticos u otros símbolos, debe entrarlos como parte de la sintaxis.

A veces, una sola variable representa un conjunto de varios parámetros. Por ejemplo, en el diagrama siguiente, la variable bloque-parámetros se puede sustituir por cualquiera de las interpretaciones del diagrama cuya cabecera es **bloque-parámetros**:



bloque-parámetros:



Los segmentos adyacentes que aparecen entre "puntos grandes" (●) se pueden especificar en cualquier secuencia.



El diagrama anterior muestra que el elemento2 y el elemento3 se pueden especificar en cualquier orden. Son válidos los dos diagramas siguientes:

SENTENCIA elemento1 elemento2 elemento3 elemento4
 SENTENCIA elemento1 elemento3 elemento2 elemento4

Convenios utilizados en este manual

Esta sección especifica algunos convenios que se utilizan coherentemente en este manual.

Condiciones de error

Una condición de error se indica en el texto del manual listando entre paréntesis el SQLSTATE asociado al error. Por ejemplo: Una signatura duplicada genera un error SQL (SQLSTATE 42723).

Convenios de resaltado

Se utilizan los siguientes convenios en este manual.

Negrita	Indica mandatos, palabras clave y otros elementos cuyos nombres están predefinidos por el sistema.
<i>Cursiva</i>	Indica una de las siguientes circunstancias: <ul style="list-style-type: none">• Nombres o valores (variables) que debe suministrar el usuario• Énfasis general• La presentación de un término nuevo• Una referencia a otra fuente de información.
Monoespaciado	Indica una de las siguientes circunstancias: <ul style="list-style-type: none">• Archivos o directorios• Información que se indica al usuario que escriba en un indicador de mandatos o en una ventana.• Ejemplos de valores de datos específicos• Ejemplos de texto similar a lo que puede mostrar el sistema• Ejemplos de mensajes del sistema.

Documentación relacionada para este manual

Las siguientes publicaciones pueden ser útiles en la preparación de aplicaciones:

- *Administration Guide*
 - Contiene la información necesaria para diseñar, implantar y mantener una base de datos a la que se va a acceder de forma local o en un entorno de cliente/servidor.
- *Application Development Guide*
 - Explica el proceso de desarrollo de aplicaciones y la forma de codificar, compilar y ejecutar programas de aplicación que utilizan SQL intercalado y API para acceder a la base de datos.
- *DB2 Universal Database for iSeries SQL Reference*
 - Este manual define el Lenguaje de consulta estructurada (SQL) soportado por DB2 Query Manager y SQL Development Kit en iSeries (AS/400). Contiene información de consulta para las tareas de administración del sistema, administración de la base de datos, programación de

aplicaciones y operación. Este manual incluye sintaxis, notas acerca del uso, palabras claves y ejemplos para cada una de las sentencias de SQL utilizadas en sistemas iSeries (AS/400) que ejecutan DB2.

- *DB2 Universal Database for OS/390 and z/OS SQL Reference*
 - Este manual define el Lenguaje de consulta estructurada (SQL) utilizado en DB2 para z/OS (OS/390). Este manual proporciona formularios de consulta, sentencias de SQL, sentencias de procedimientos de SQL, límites de DB2, SQLCA, SQLDA, tablas de catálogos y palabras reservadas de SQL para sistemas z/OS (OS/390) que ejecutan DB2.
- *DB2 Spatial Extender Guía del usuario y de consulta*
 - Describe cómo escribir aplicaciones para crear y utilizar un sistema de información geográfica (geographic information system, GIS). Para crear y utilizar un GIS es necesario proporcionar una base de datos con recursos y luego consultar los datos para obtener información, tal como ubicaciones, distancias y distribuciones dentro de zonas geográficas.
- *Consulta de SQL de IBM*
 - Este manual contiene todos los elementos comunes del SQL que están distribuidos por toda la biblioteca IBM de productos de base de datos. Proporciona límites y normas que pueden servir de ayuda en la preparación de programas portables que utilicen bases de datos IBM. Proporciona una lista de extensiones SQL e incompatibilidades entre los siguientes estándares y productos: SQL92E, XPG4-SQL, IBM-SQL y los productos de base de datos relacionales IBM.
- *American National Standard X3.135-1992, Database Language SQL*
 - Contiene la definición estándar ANSI del SQL.
- *ISO/IEC 9075:1992, Database Language SQL*
 - Contiene la definición del SQL proporcionada por la norma 1992 de ISO.
- *ISO/IEC 9075-2:1999, Database Language SQL -- Part 2: Foundation (SQL/Foundation)*
 - Contiene una gran parte de la definición del SQL proporcionada por la norma 1999 de ISO.
- *ISO/IEC 9075-4:1999, Database Language SQL -- Part 4: Persistent Stored Modules (SQL/PSM)*
 - Contiene la definición de las sentencias de control de los procedimientos SQL, tal como aparece en la norma 1999 de ISO.
- *ISO/IEC 9075-5:1999, Database Language SQL -- Part 4: Host Language Bindings (SQL/Bindings)*
 - Contiene la definición de los enlaces de lenguaje principal y del SQL dinámico, tal como aparece en la norma 1999 de ISO.

Capítulo 2. Conceptos

Este capítulo proporciona una visión general de los conceptos utilizados más comúnmente en el Lenguaje de consulta estructurada (SQL). La intención de este capítulo es proporcionar una visión de alto nivel de los conceptos. La información de consulta que le sigue proporciona una visión más detallada.

Definiciones de base de datos relacional

Una *base de datos relacional* es una base de datos que se trata como un conjunto de tablas y se manipula de acuerdo con el modelo de datos relacional. Contiene un conjunto de objetos que se utilizan para almacenar y gestionar los datos, así como para acceder a los mismos. Las tablas, vistas, índices, funciones, desencadenantes y paquetes son ejemplos de estos objetos.

Una base de datos relacional *particionada* es una base de datos relacional en la que los datos se gestionan repartidos en múltiples particiones (también denominadas nodos). Esta separación de los datos entre particiones es transparente para los usuarios de la mayoría de sentencias de SQL. Sin embargo, algunas sentencias DDL (Lenguaje de definición de datos)¹ tienen en cuenta la información sobre particiones (por ejemplo, CREATE NODEGROUP).

Una base de datos *federada* es una base de datos relacional en la que los datos están almacenados en varias fuentes de datos (tales como bases de datos relacionales separadas). Los datos son tratados como si pertenecieran a una sola gran base de datos y se pueden acceder mediante las consultas SQL normales. Los cambios en los datos se pueden dirigir explícitamente hacia la fuente de datos apropiada. Vea “Sistemas federados DB2” en la página 56 para obtener más información.

Lenguaje de consulta estructurada (SQL)

SQL es un lenguaje estandarizado que sirve para definir y manipular los datos de una base de datos relacional. De acuerdo con el modelo relacional de datos, la base de datos se crea como un conjunto de tablas, las relaciones se representan mediante valores en las tablas y los datos se recuperan especificando una tabla de resultados que puede derivarse de una o más tablas base.

1. Lenguaje de definición de datos (Data Definition Language - DDL) es el subconjunto de sentencias de SQL utilizadas para describir las relaciones de los datos en una base de datos.

Las sentencias de SQL las ejecuta un gestor de bases de datos. Una de las funciones del gestor de bases de datos es transformar la especificación de una tabla resultante en una secuencia de operaciones internas que optimicen la recuperación de datos. Esta transformación se produce en dos fases: preparación y enlace lógico.

Todas las sentencias de SQL ejecutables deben prepararse antes de su ejecución. El resultado de esta preparación es el formato operativo o ejecutable de la sentencia. El método de preparación de una sentencia de SQL y la persistencia de su formato operativo distinguen el SQL estático del SQL dinámico.

Autorización y privilegios

Una *autorización* permite a un usuario o grupo realizar una tarea general como, por ejemplo, la conexión a una base de datos, la creación de tablas o la administración de un sistema. Un *privilegio* da a un usuario o a un grupo el derecho a acceder a un objeto específico de la base de datos de una manera especificada.

El gestor de bases de datos necesita que un usuario esté específicamente autorizado, ya sea implícita o explícitamente, para utilizar cada función de base de datos necesaria para que ese usuario realice una tarea específica. Las autorizaciones o los privilegios *explícitos* se otorgan al usuario (GRANTEETYPE de U). Las autorizaciones o los privilegios *implícitos* se otorgan a un grupo al que pertenece el usuario (GRANTEETYPE de G). De este modo, para crear una tabla, un usuario debe tener autorización para crear tablas; para modificar una tabla, un usuario debe tener autorización para modificar la tabla y así sucesivamente.

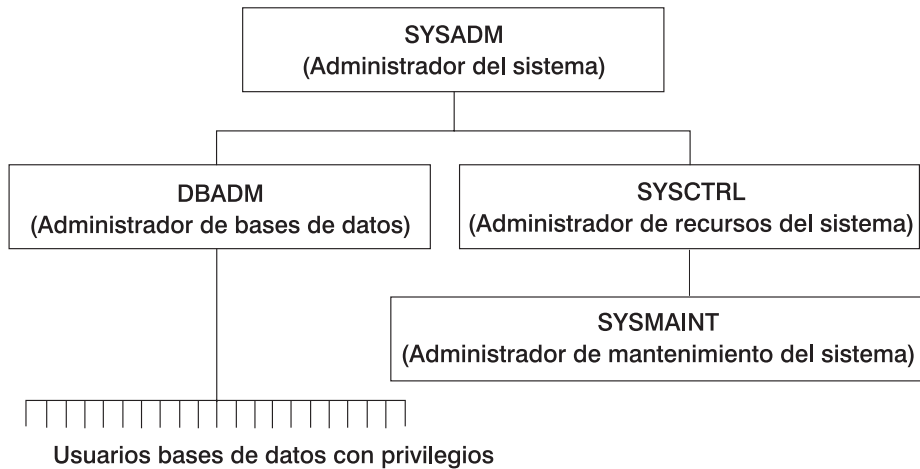


Figura 1. Jerarquía de autorizaciones y privilegios

La persona o personas con autorización de administrador se encargan de la tarea de controlar el gestor de bases de datos y son responsables de la seguridad e integridad de los datos. Controlan quién va a tener acceso al gestor de bases de datos y hasta qué punto tiene acceso cada usuario.

El gestor de bases de datos proporciona dos autorizaciones de administración:

- SYSADM - autorización de administrador del sistema
- DBADM - autorización de administrador de bases de datos

El gestor de bases de datos también proporciona dos autorizaciones de control del sistema:

- SYSCTRL - autorización de control del sistema
- SYSMAINT - autorización de mantenimiento del sistema

La autorización SYSADM es el nivel más alto de autorización y tiene control sobre todos los recursos que crea y mantiene el gestor de bases de datos. La autorización SYSADM incluye todos los privilegios de DBADM, SYSCTRL y SYSMAINT, así como la autorización para otorgar o revocar autorizaciones DBADM.

La autorización DBADM es la autorización de administración específica para una sola base de datos. Esta autorización incluye privilegios para crear objetos, emitir mandatos de la base de datos y acceder a los datos de sus tablas mediante sentencias de SQL. La autorización DBADM también incluye la autorización para otorgar o revocar el privilegio CONTROL y privilegios individuales.

La autorización SYSCTRL corresponde al nivel más alto de autorización de control del sistema y sólo se aplica a las operaciones que afectan a los recursos del sistema. No permite el acceso directo a los datos. Esta autorización incluye privilegios para crear, actualizar o eliminar una base de datos, detener una instancia o base de datos y eliminar o crear un espacio de tablas.

La autorización SYSMANT es el segundo nivel de autorización de control del sistema. Un usuario con autorización SYSMANT puede realizar operaciones de mantenimiento en todas las bases de datos asociadas a una instancia. No permite el acceso directo a los datos. Esta autorización incluye privilegios para actualizar archivos de configuración de la base de datos, realizar una copia de seguridad de una base de datos o un espacio de tablas, restaurar una base de datos existente y supervisar una base de datos.

Las autorizaciones de base de datos se aplican a aquellas actividades que un administrador ha permitido realizar a un usuario que no se aplican a una instancia específica de un objeto de base de datos. Por ejemplo, puede otorgarse a un usuario la autorización para crear paquetes pero no para crear tablas.

Los privilegios se aplican a aquellas actividades que un administrador o un propietario de un objeto han permitido que un usuario pudiese realizar en objetos de la base de datos. Los usuarios con privilegios pueden crear objetos, aunque se les aplican algunas restricciones, a diferencia de un usuario con una autorización como SYSADM o DBADM. Por ejemplo, un usuario puede tener el privilegio para crear una vista en una tabla pero no un desencadenante en la misma tabla. Los usuarios con privilegios tienen acceso a los objetos de los que son propietarios y pueden transmitir privilegios sobre sus propios objetos a otros usuarios utilizando la sentencia GRANT.

El privilegio CONTROL permite al usuario acceder a un objeto de base de datos específico del modo necesario y otorgar (GRANT) y revocar (REVOKE) privilegios a otros usuarios sobre ese objeto. Para conceder el privilegio CONTROL se necesita la autorización DBADM.

Los privilegios individuales y las autorizaciones de bases de datos permiten una función específica pero no incluyen el derecho a otorgar los mismos privilegios o autorizaciones a otros usuarios. El derecho para otorgar privilegios de tabla, vista o esquema a otros puede ampliarse a otros usuarios utilizando WITH GRANT OPTION en la sentencia GRANT.

Esquemas

Un *esquema* es un conjunto de objetos con nombre. Los esquemas proporcionan una clasificación lógica de los objetos de la base de datos. Un esquema puede contener tablas, vistas, apodos, desencadenantes, funciones, paquetes y otros objetos.

Un esquema también es un objeto en la base de datos. Se crea explícitamente utilizando la sentencia `CREATE SCHEMA` con el usuario actual registrado como propietario del esquema. También se puede crear implícitamente cuando se crea otro objeto, a condición de que el usuario tenga la autorización `IMPLICIT_SCHEMA`.

El *nombre de esquema* se utiliza como la parte más a la izquierda de un nombre de objeto de dos partes. Si el objeto se califica específicamente con un nombre de esquema al crearse, se asigna el objeto a dicho esquema. Si no se especifica ningún nombre de esquema al crear el objeto, se utiliza el nombre de esquema por omisión.

Por ejemplo, un usuario con autorización `DBADM` crea un esquema llamado `C` para el usuario `A`:

```
CREATE SCHEMA C AUTHORIZATION A
```

El usuario `A` puede emitir la siguiente sentencia para crear una tabla llamada `X` en el esquema `C`:

```
CREATE TABLE C.X (COL1 INT)
```

Control de la utilización de esquemas

Cuando se crea una base de datos, todos los usuarios tienen la autorización `IMPLICIT_SCHEMA`. Esto permite a cualquier usuario crear objetos en cualquier esquema que aún no exista. Un esquema creado implícitamente permite a cualquier usuario crear otros objetos en dicho esquema. La posibilidad de crear seudónimos, tipos diferenciados, funciones y desencadenantes se amplía a los esquemas creados implícitamente. Los privilegios por omisión de un esquema creado implícitamente proporcionan compatibilidad con las versiones anteriores.

Si se revoca la autorización `IMPLICIT_SCHEMA` de `PUBLIC`, los esquemas se crean explícitamente utilizando la sentencia `CREATE SCHEMA` o los crean implícitamente los usuarios (por ejemplo, los que tienen autorización `DBADM`) a los que se otorga la autorización `IMPLICIT_SCHEMA`. Aunque la revocación de la autorización `IMPLICIT_SCHEMA` de `PUBLIC` incrementa el control sobre la utilización de los nombres de esquema, también puede producir errores de autorización en aplicaciones existentes cuando estas aplicaciones intentan crear objetos.

Los esquemas también tienen privilegios asociados, que permiten al propietario del esquema controlar qué usuarios tienen privilegio para crear, modificar y eliminar objetos del esquema. A un propietario de esquema se le dan inicialmente todos estos privilegios en el esquema, con la posibilidad de otorgarlos a otros usuarios. Un esquema creado implícitamente es de propiedad del sistema y a todos los usuarios se les proporciona inicialmente el privilegio de crear objetos en dicho esquema. Un usuario con las autorizaciones DBADM o SYSADM puede cambiar los privilegios que poseen los usuarios en cualquier esquema. Por consiguiente, se puede controlar el acceso para crear, modificar y eliminar objetos en cualquier esquema (incluso uno creado implícitamente).

Tablas

Las tablas son estructuras lógicas mantenidas por el gestor de bases de datos. Las tablas están formadas por columnas y filas. Las filas de una tabla no están necesariamente ordenadas (el orden lo determina el programa de aplicación). En la intersección de cada columna con una fila hay un elemento de datos específico denominado *valor*. Una *columna* es un conjunto de valores del mismo tipo o de uno de sus subtipos. Una *fila* es una secuencia de valores ordenados de forma que el valor *n* sea el valor de la columna *n* de la tabla.

Una *tabla base* se crea con la sentencia CREATE TABLE y se utiliza para conservar los datos habituales de los usuarios. Una *tabla resultante* es un conjunto de filas que el gestor de bases de datos selecciona o genera a partir de una o varias tablas base para satisfacer una consulta.

Una *tabla de resumen* es una tabla definida por una consulta que se utiliza también para determinar los datos de la tabla. Las tablas de resumen se pueden utilizar para mejorar el rendimiento de las consultas. Si el gestor de bases de datos determina que se puede resolver una parte de una consulta utilizando una tabla de resumen, el gestor de bases de datos puede volver a escribir la consulta para utilizar la tabla de resumen. Esta decisión se basa en los valores de configuración de base de datos tales como los registros especiales CURRENT REFRESH AGE y CURRENT QUERY OPTIMIZATION.

Una tabla puede definir el tipo de datos de cada columna por separado o basar los tipos de las columnas en los atributos de un tipo estructurado definido por el usuario. Esto se denomina *tabla con tipo*. Un tipo estructurado definido por el usuario puede formar parte de una jerarquía de tipos. Un *subtipo* hereda los atributos de su *supertipo*. De manera similar, una tabla con tipo puede formar parte de una jerarquía de tablas. Una *subtabla* hereda las columnas de su *supertabla*. Tenga en cuenta que el término *subtipo* se aplica a un tipo estructurado definido por el usuario y a todos los tipos estructurados definidos por el usuario que están por debajo del mismo en la jerarquía de tipos. Un *subtipo correspondiente* de un tipo estructurado T es un tipo

estructurado por debajo de T en la jerarquía de tipos. De forma similar, el término *subtabla* se aplica a una tabla con tipo y a todas las tablas con tipo que están por debajo de la misma en la jerarquía de tablas. Una *subtabla correspondiente* de una tabla T es una tabla que está por debajo de T en la jerarquía de tablas.

Una *tabla temporal declarada* se crea mediante una sentencia DECLARE GLOBAL TEMPORARY TABLE y se utiliza para contener datos temporales para una aplicación individual. Esta tabla se elimina implícitamente cuando la aplicación se desconecta de la base de datos.

Vistas

Una *vista* proporciona una manera alternativa de consultar los datos de una o varias tablas.

Una vista es una especificación de una tabla resultante a la que se le da un nombre. La especificación es una sentencia SELECT que se ejecuta siempre que se hace referencia a la vista en una sentencia de SQL. Puede considerarse que una vista tiene columnas y filas igual que una tabla base. Para efectuar una recuperación, se pueden utilizar todas las vistas como si fueran tablas base. El hecho de que una vista pueda utilizarse en una operación de inserción, actualización o supresión dependerá de su definición, tal como se explica en la descripción de la sentencia CREATE VIEW. Vea "CREATE VIEW" en la página 957 para obtener más información.

Cuando la columna de una vista se obtiene directamente de una columna de una tabla base, esa columna hereda todas las restricciones aplicables a la columna de la tabla base. Por ejemplo, si una vista incluye una clave foránea de su tabla base, las operaciones INSERT y UPDATE que utilicen dicha vista estarán sujetas a las mismas restricciones de referencia que la tabla base. Asimismo, si la tabla base de una vista es una tabla padre, las operaciones DELETE y UPDATE que utilicen dicha vista estarán sujetas a las mismas reglas que las operaciones DELETE y UPDATE de la tabla base.

Una vista puede obtener el tipo de datos de cada columna de la tabla resultante o basar los tipos para las columnas en los atributos de un tipo estructurado definido por el usuario. Esta vista se denomina *vista con tipo*. De manera similar a una tabla con tipo, una vista con tipo puede formar parte de una jerarquía de vistas. Una *subvista* hereda las columnas de su *supervista*. El término *subvista* se aplica a una vista con tipo y a todas las vistas con tipo que están por debajo de la misma en la jerarquía de vistas. Una *subvista correspondiente* de una vista V es una vista por debajo de V en la jerarquía de vistas con tipo.

Una vista puede quedar no operativa (por ejemplo, si se elimina la tabla base); si ocurre esto, la vista ya no estará disponible para las sentencias de SQL.

Seudónimos

Un *seudónimo* es un nombre alternativo para una tabla o una vista. Se puede utilizar para hacer referencia a una tabla o vista en aquellos casos en los que se puede hacer referencia a una tabla o vista existente. Una *seudónimo* no puede utilizarse en todos los contextos. Por ejemplo, no puede utilizarse en la condición de comprobación de una restricción de comprobación. Un *seudónimo* tampoco puede hacer referencia a una tabla temporal declarada.

Al igual que las tablas y las vistas, un *seudónimo* puede crearse, eliminarse y tener comentarios asociados. Sin embargo, a diferencia de las tablas, los *seudónimos* pueden hacerse referencia entre sí en un proceso llamado *encadenamiento*. Los *seudónimos* son nombres referidos públicamente, por lo que no es necesaria ninguna autorización ni privilegio especial para utilizarlos. Sin embargo, para acceder a las tablas y vistas a las que hace referencia el *seudónimo* es necesario tener la autorización apropiada para el contexto actual.

Además de los *seudónimos* de la tabla, existen otros tipos de *seudónimos* como, por ejemplo, los *seudónimos* de base de datos y los de red. También se pueden crear *seudónimos* para los *apodos* (vistas o tablas de datos ubicadas en sistemas federados).

Consulte el apartado “*Seudónimos*” en la página 78 y el apartado “*CREATE ALIAS*” en la página 670 para obtener más información sobre los *seudónimos*.

Índices

Un *índice* es un conjunto ordenado de punteros para filas de una tabla base. Cada *índice* se basa en los valores de los datos de una o varias columnas de la tabla. Un *índice* es un objeto que está separado de los datos de la tabla. Cuando se crea un *índice*, el gestor de bases de datos crea esta estructura y la mantiene automáticamente.

El gestor de bases de datos utiliza los *índices* para:

- Mejorar el rendimiento. En la mayoría de los casos, el acceso a los datos es más rápido que sin *índices*.

No puede crearse un *índice* para una vista. Sin embargo, un *índice* creado para una tabla en la que se basa una vista puede mejorar a veces el rendimiento de las operaciones en la vista.

- Asegurar la exclusividad. Una tabla con un índice de unicidad no puede tener filas con claves idénticas.

Claves

Una *clave* es un conjunto de columnas que se pueden utilizar para identificar o para acceder a una fila o filas determinadas. La clave viene identificada en la descripción de una tabla, índice o restricción de referencia. Una misma columna puede formar parte de más de una clave.

Una clave compuesta de más de una columna se denomina una *clave compuesta*. En una tabla con una clave compuesta, el orden de las columnas dentro de la clave compuesta no está restringido por el orden de las columnas en la tabla. El *valor* de una clave compuesta indica un valor compuesto. Así, por ejemplo la regla “el valor de la clave foránea debe ser igual al valor de la clave primaria” significa que cada componente del valor de la clave foránea debe ser igual al componente del valor correspondiente de la clave primaria.

Claves de unicidad

Una *clave de unicidad* es una clave restringida de manera que no puede tener dos valores iguales. Las columnas de una clave de unicidad no pueden contener valores nulos. El gestor de bases de datos impone la restricción durante la ejecución de cualquier operación que cambie los valores de los datos como, por ejemplo, INSERT o UPDATE. El mecanismo utilizado para imponer la restricción se denomina *índice de unicidad*. De este modo, cada clave de unicidad es una clave de un índice de unicidad. También se dice que dichos índices tienen el atributo UNIQUE. Vea “Restricciones de unicidad” en la página 18 para obtener una descripción más detallada.

Claves primarias

Una *clave primaria* es un caso especial de clave de unicidad. Una tabla no puede tener más de una clave primaria. Consulte el apartado “Claves de unicidad” para obtener una descripción más detallada.

Claves foráneas

Una *clave foránea* es una clave que se especifica en la definición de una restricción de referencia. Consulte el apartado “Restricciones de referencia” en la página 19 para obtener una descripción más detallada.

Claves de particionamiento

Una *clave de particionamiento* es una clave que forma parte de la definición de una tabla de una base de datos particionada. La clave de particionamiento se utiliza para determinar la partición en la que se almacena la fila de datos. Si se define una clave de particionamiento, las claves de unicidad y las claves primarias deben incluir las mismas columnas que la clave de particionamiento, pero pueden tener columnas adicionales. Una tabla no puede tener más de una clave de particionamiento.

Restricciones

Una *restricción* es una regla que impone el gestor de bases de datos.

Hay tres tipos de restricciones:

- Una *restricción de unicidad* es una regla que prohíbe los valores duplicados en una o varias columnas de una tabla. Las restricciones de unicidad a las que se da soporte son la clave de unicidad y la clave primaria. Por ejemplo, se puede definir una restricción de unicidad en el identificador de proveedor de la tabla de proveedores para asegurarse de que no se da el mismo identificador de proveedor a dos proveedores.
- Una *restricción de referencia* es una regla lógica acerca de los valores de una o varias columnas de una o varias tablas. Por ejemplo, un conjunto de tablas que comparte información sobre los proveedores de una empresa. Ocasionalmente, el nombre de un proveedor cambia. Puede definir una restricción de referencia que indique que el ID del proveedor de una tabla debe coincidir con un ID de proveedor de la información de proveedor. Esta restricción impide que se realicen inserciones, actualizaciones o supresiones, que de lo contrario, harán que falte información del proveedor.
- Una *restricción de comprobación de tabla* establece restricciones en los datos que se añaden a una tabla específica. Por ejemplo, la restricción puede restringir el nivel salarial de un empleado para que no sea inferior a 70.000 pts siempre que se añadan o se actualicen datos salariales en una tabla que contiene información de personal.

Las restricciones de referencia y de comprobación de tabla pueden activarse y desactivarse. Normalmente, la aplicación de una restricción se desactiva cuando se cargan grandes cantidades de datos en la base de datos. Los detalles sobre la activación o desactivación de las restricciones se describen en el apartado “SET INTEGRITY” en la página 1168.

Restricciones de unicidad

Una *restricción de unicidad* es la regla que establece que los valores de una clave sólo son válidos si son exclusivos en una tabla. Las restricciones de unicidad son opcionales y pueden definirse en las sentencias CREATE TABLE o ALTER TABLE utilizando la cláusula PRIMARY KEY o la cláusula UNIQUE. Las columnas especificadas en una restricción de unicidad deben definirse como NOT NULL. El gestor de bases de datos utiliza un índice de unicidad para forzar la unicidad de la clave durante los cambios en las columnas de la restricción de unicidad.

Una tabla puede tener un número arbitrario de restricciones de unicidad y como máximo una restricción de unicidad definida como la clave primaria. Una tabla no puede tener más de una restricción de unicidad en el mismo conjunto de columnas.

Una restricción de unicidad a la que hace referencia la clave foránea de una restricción de referencia se denomina *clave padre*.

Cuando se define una restricción de unicidad en una sentencia CREATE TABLE, el gestor de bases de datos crea automáticamente un índice de unicidad y lo designa como un índice principal o de unicidad necesario para el sistema.

Cuando se define una restricción de unicidad en una sentencia ALTER TABLE y existe un índice en las mismas columnas, dicho índice se designa como de unicidad y necesario para el sistema. Si no existe tal índice, el gestor de bases de datos crea automáticamente el índice de unicidad y lo designa como un índice principal o de unicidad necesario para el sistema.

Observe que existe una distinción entre la definición de una restricción de unicidad y la creación de un índice de unicidad. Aunque ambos impongan la exclusividad, un índice de unicidad permite la existencia de columnas que pueden contener valores nulos y generalmente no puede utilizarse como una clave padre.

Restricciones de referencia

La *integridad de referencia* es el estado de una base de datos en la que todos los valores de todas las claves foráneas son válidos. Una *clave foránea* es una columna o un conjunto de columnas de una tabla cuyos valores deben coincidir obligatoriamente con, como mínimo, un valor de una clave primaria o de una clave de unicidad de una fila de su tabla padre. Una *restricción de referencia* es la regla que establece que los valores de la clave foránea sólo son válidos si se cumple una de estas condiciones:

- Aparecen como valores de una clave padre.
- Algún componente de la clave foránea es nulo.

La tabla que contiene la clave padre se denomina la *tabla padre* de la restricción de referencia y se dice que la tabla que contiene la clave foránea es *dependiente* de dicha tabla.

Las restricciones de referencia son opcionales y pueden definirse en sentencias CREATE TABLE y ALTER TABLE. Las restricciones de referencia las impone el gestor de bases de datos durante la ejecución de las sentencias INSERT, UPDATE, DELETE, ALTER TABLE, ADD CONSTRAINT y SET INTEGRITY. Esta imposición se hace efectiva al finalizar la sentencia.

Las restricciones de referencia con una regla de supresión o actualización de RESTRICT se imponen antes que el resto de restricciones de referencia. Las restricciones de referencia con una regla de supresión o actualización de NO

ACTION tienen un funcionamiento igual que RESTRICT, en la mayoría de casos. Sin embargo, en determinadas sentencias de SQL pueden existir diferencias.

Tenga en cuenta que la integridad de referencia, las restricciones de comprobación y los desencadenantes pueden combinarse en la ejecución. Para obtener más información sobre la combinación de estos tres elementos, consulte el “Apéndice J. Interacción de desencadenantes y restricciones” en la página 1455.

En las reglas de integridad de referencia se utilizan los conceptos y terminología siguientes:

Clave padre

Clave primaria o clave de unicidad de una restricción de referencia.

Fila padre

Fila que tiene, como mínimo, una fila dependiente.

Tabla padre

Tabla que contiene la clave padre de una restricción de referencia. Una tabla puede definirse como padre en un número arbitrario de restricciones de referencia. Una tabla que es padre en una restricción de referencia también puede ser dependiente de una restricción de referencia.

Tabla dependiente

Tabla que contiene como mínimo una restricción de referencia en su definición. Una tabla puede definirse como dependiente en un número arbitrario de restricciones de referencia. Una tabla que es dependiente en una restricción de referencia también puede ser padre de una restricción de referencia.

Tabla descendiente

Una tabla es descendiente de la tabla T si es dependiente de T o descendiente de una tabla dependiente de T.

Fila dependiente

Fila que tiene, como mínimo, una fila padre.

Fila descendiente

Una fila es descendiente de la fila r si es dependiente de r o descendiente de una dependiente de r.

Ciclo de referencia

Conjunto de restricciones de referencia en el que cada tabla del conjunto es descendiente de sí misma.

Fila de autorreferencia

Fila que es padre de sí misma.

Tabla autorreferente

Tabla que es padre y dependiente en la misma restricción de referencia. La restricción se denomina *restricción de autorreferencia*.

Regla de inserción

La regla de inserción de una restricción de referencia es la que establece que un valor de inserción que no sea nulo de la clave foránea debe coincidir con algún valor de la clave padre de la tabla padre. El valor de la clave foránea compuesta será nulo si algún componente del valor es nulo. Es una regla implícita cuando se especifica una clave foránea.

Regla de actualización

La regla de actualización de una restricción de referencia se especifica al definir la restricción de referencia. Las opciones son NO ACTION y RESTRICT. La regla de actualización se aplica al actualizar una fila de la tabla padre o una fila de la tabla dependiente.

En el caso de una fila padre, cuando se actualiza un valor de una columna de la clave padre, se aplican estas reglas:

- Si cualquier fila de la tabla dependiente coincide con el valor original de la clave, se rechaza la actualización cuando la regla de actualización es RESTRICT.
- Si cualquier fila de la tabla dependiente no tiene una clave padre correspondiente cuando se completa la sentencia de actualización (excluyendo los desencadenantes AFTER), se rechaza la actualización cuando la regla de actualización es NO ACTION.

En el caso de una fila dependiente, la regla de actualización NO ACTION es implícita cuando se especifica una clave foránea. NO ACTION significa que un valor de actualización que no sea nulo de una clave foránea debe coincidir con algún valor de la clave padre de la tabla padre cuando se complete la sentencia de actualización.

El valor de la clave foránea compuesta será nulo si algún componente del valor es nulo.

Regla de supresión

La regla de supresión de una restricción de referencia se especifica al definir la restricción de referencia. Las opciones son NO ACTION, RESTRICT, CASCADE o SET NULL. SET NULL sólo puede especificarse si hay alguna columna de la clave foránea que permita valores nulos.

La regla de supresión de una restricción de referencia se aplica al suprimir una fila de la tabla padre. Para ser más exactos, esta regla se aplica cuando una fila de la tabla padre es el objeto de una operación de supresión o de supresión propagada (definida a continuación) y dicha fila tiene dependientes

en la tabla dependiente de la restricción de referencia. Examinemos un ejemplo donde P es la tabla padre, D es la tabla dependiente y p es una fila padre que es el objeto de una operación de supresión o de supresión propagada. La regla de supresión funciona del modo siguiente:

- Para RESTRICT o NO ACTION, se produce un error y no se suprime ninguna fila.
- Para CASCADE, la operación de supresión se propaga a los dependientes de p en la tabla D.
- Para SET NULL, cada columna con posibilidad de nulos de la clave foránea de cada dependiente de p en la tabla D se establece en nulo.

Cada restricción de referencia en la que una tabla sea padre tiene su propia regla de supresión, y todas las reglas de supresión aplicables se utilizan para determinar el resultado de la operación de supresión. Así, una fila no puede suprimirse si tiene dependientes en una restricción de referencia con una regla de supresión RESTRICT o NO ACTION o la supresión se propaga en cascada a cualquiera de sus descendientes que sean dependientes en una restricción de referencia con la regla de supresión RESTRICT o NO ACTION.

La supresión de una fila de la tabla padre P implica a otras tablas y puede afectar a las filas de dichas tablas:

- Si la tabla D es dependiente de P y la regla de supresión es RESTRICT o NO ACTION, D se implicará en la operación pero no se verá afectada por la misma.
- Si D es dependiente de P y la regla de supresión es SET NULL, D estará implicada en la operación y las filas de D podrán actualizarse durante la operación.
- Si D es dependiente de P y la regla de supresión es CASCADE, D estará implicada en la operación y las filas de D podrán suprimirse durante la operación.

Si se suprimen filas de D, se dice que la operación de supresión en P se propaga a D. Si D es también una tabla padre, las acciones descritas en esta lista se aplican a su vez a los elementos dependientes de D.

De cualquier tabla que pueda estar implicada en una operación de supresión en P se dice que está *conectada por supresión* a P. Así, una tabla está conectada por supresión a la tabla P si es dependiente de P o es dependiente de una tabla hacia la que se propagan en cascada operaciones de supresión desde P.

Restricciones de comprobación de tabla

Una *restricción de comprobación de tabla* es una regla que especifica los valores permitidos en una o varias columnas de cada fila de una tabla. Una restricción es opcional y puede definirse utilizando las sentencias de SQL CREATE TABLE y ALTER TABLE. La especificación de restricciones de

comprobación de tabla se realiza mediante una forma restringida de condición de búsqueda. Una de las restricciones consiste en que un nombre de columna de una restricción de comprobación de tabla de la tabla *T* debe identificar una columna de *T*.

Una tabla puede tener un número arbitrario de restricciones de comprobación de tabla. Éstas se imponen cuando se inserta una fila en la tabla o se actualiza una fila de la tabla.

Una restricción de comprobación de tabla se impone aplicando su condición de búsqueda en cada fila que se inserte o actualice. Si el resultado de la condición de búsqueda es falso en cualquiera de las filas, se produce un error.

Cuando hay una o varias restricciones de comprobación de tabla definidas en la sentencia ALTER TABLE para una tabla en la que existen datos, éstos se comprueban con la nueva condición antes de que se complete satisfactoriamente la sentencia ALTER TABLE. La tabla puede ponerse en *estado pendiente de comprobación* lo que permitirá que la sentencia ALTER TABLE se realice satisfactoriamente sin tener que comprobar los datos. La sentencia SET INTEGRITY se utiliza para poner la tabla en estado pendiente de comprobación. También se utiliza para reanudar la comparación de cada fila con la restricción.

Nivel de aislamiento

El *nivel de aislamiento* asociado con un proceso de aplicación define el grado de aislamiento de dicho proceso de aplicación respecto a otros procesos de aplicación que se ejecutan concurrentemente. Por consiguiente, el nivel de aislamiento de un proceso de aplicación específica:

- El grado al que las filas leídas y actualizadas por la aplicación están disponibles para otros procesos de aplicación que se ejecutan concurrentemente.
- El grado al que puede afectar a la aplicación la actividad de actualización de otros procesos de aplicación que se ejecutan concurrentemente.

El nivel de aislamiento se especifica como un atributo de un paquete y se aplica a los procesos de aplicación que hacen uso del paquete. El nivel de aislamiento se especifica en el proceso de preparación del proceso. En función del tipo de bloqueo, limita o impide el acceso a los datos por parte de procesos de aplicación concurrentes. Para obtener detalles sobre los diferentes tipos y atributos de bloqueos específicos, consulte la publicación *Administration Guide*. Las tablas temporales declaradas y sus filas no se bloquean, pues sólo pueden ser accedidas por la aplicación que declaró las tablas temporales. Por lo tanto, la información siguiente sobre el bloqueo y los niveles de aislamiento no es aplicable a las tablas temporales declaradas.

El gestor de bases de datos da soporte a tres categorías generales de bloqueos:

Compartimiento

Limita los procesos de aplicación concurrentes a operaciones de sólo lectura de los datos.

Actualización

Limita los procesos de aplicación concurrentes a operaciones de sólo lectura de los datos, siempre que dichos procesos no hayan declarado que pueden actualizar la fila. El gestor de bases de datos supone que el proceso que consulta actualmente la fila puede actualizarla.

Exclusividad

Evita que los procesos de aplicación concurrentes accedan a los datos de cualquier modo, excepto los procesos de aplicación con un nivel de aislamiento de *lectura no confirmada*, que pueden leer los datos pero no modificarlos. (Consulte el apartado “Lectura no confirmada (UR)” en la página 26.)

El bloqueo se produce en la fila de la tabla base. Sin embargo, el gestor de bases de datos puede sustituir múltiples bloqueos de filas por un solo bloqueo de tabla. Esto se denomina *escalada de bloqueos*. Un proceso de aplicación tiene garantizado al menos el nivel mínimo de bloqueo solicitado.

El gestor de bases de datos de DB2 Universal Database da soporte a cuatro niveles de aislamiento. Independientemente del nivel de aislamiento, el gestor de bases de datos coloca bloqueos de exclusividad en cada fila que se inserta, actualiza o suprime. Por lo tanto, los niveles de aislamiento aseguran que las filas que cambia el proceso de aplicación durante una unidad de trabajo no las pueda modificar ningún otro proceso de aplicación hasta que la unidad de trabajo haya finalizado. Los niveles de aislamiento son:

Lectura repetible (RR)

El nivel de Lectura repetible asegura que:

- Cualquier fila leída durante una unidad de trabajo² no sea modificada por otros procesos de aplicación hasta que la unidad de trabajo se haya completado.³
- Las filas modificadas por otro proceso de aplicación no se pueden leer hasta que dicho proceso de aplicación las confirme.

2. Las filas se leen en la misma unidad de trabajo que la sentencia OPEN correspondiente. Consulte WITH HOLD en el apartado “DECLARE CURSOR” en la página 976.

3. El uso de la cláusula WITH RELEASE opcional en la sentencia CLOSE significa que, si se vuelve a abrir el cursor, ya no se aplicará ninguna garantía respecto a las lecturas no repetibles y no se aplicarán ya lecturas fantasma a ninguna fila a la que se haya accedido anteriormente.

El nivel de Lectura repetible no permite ver las filas fantasma (consulte Estabilidad de lectura).

Además de los bloqueos de exclusividad, un proceso de aplicación que se ejecute en el nivel RR adquiere, como mínimo, bloqueos de compartimiento en todas las filas a las que hace referencia. Además, el bloqueo se realiza de forma que el proceso de aplicación quede completamente aislado de los efectos de los procesos de aplicación concurrentes.

Estabilidad de lectura (RS)

Igual que el nivel de Lectura repetible, el nivel de Estabilidad de lectura asegura que:

- Cualquier fila leída durante una unidad de trabajo⁴ no sea modificada por otros procesos de aplicación hasta que la unidad de trabajo se haya completado.⁵
- Las filas modificadas por otro proceso de aplicación no se pueden leer hasta que dicho proceso de aplicación las confirme.

A diferencia de la Lectura repetible, el nivel de Estabilidad de lectura no aísla completamente el proceso de aplicación de los efectos de procesos de aplicación concurrentes. En el nivel RS, los procesos de aplicación que emiten la misma consulta más de una vez pueden ver filas adicionales producidas por la adición de información nueva a la base de datos que realizan otros procesos de aplicación. Estas filas adicionales se denominan *filas fantasma*.

Por ejemplo, puede aparecer una fila fantasma en la situación siguiente:

1. El proceso de aplicación P1 lee el conjunto de filas n que satisfacen alguna condición de búsqueda.
2. Entonces el proceso de aplicación P2 inserta una o más filas que satisfacen la condición de búsqueda y confirma esas nuevas inserciones.
3. P1 lee nuevamente el conjunto de filas con la misma condición de búsqueda y obtiene tanto las filas originales como las filas insertadas por P2.

Además de los bloqueos de exclusividad, un proceso de aplicación que se ejecute al nivel RS adquiere, como mínimo, bloqueos de compartimiento en todas las filas calificadas para ello.

4. Las filas se leen en la misma unidad de trabajo que la sentencia OPEN correspondiente. Consulte WITH HOLD en el apartado "DECLARE CURSOR" en la página 976.

5. El uso de la cláusula WITH RELEASE opcional en la sentencia CLOSE significa que, si se vuelve a abrir el cursor, ya no se aplicará ninguna garantía respecto a las lecturas no repetibles a ninguna fila a la que se haya accedido anteriormente.

Estabilidad del cursor (CS)

Al igual que el nivel de Lectura repetible, la Estabilidad del cursor asegura que cualquier fila que haya sido modificada por otro proceso de aplicación no pueda leerse hasta que sea confirmada por dicho proceso de aplicación.

Sin embargo, a diferencia del nivel de Lectura repetible, la Estabilidad del cursor sólo asegura que otros procesos de aplicación no modifiquen la fila actual de cada cursor actualizable. De este modo, las filas leídas durante una unidad de trabajo pueden ser modificadas por otros procesos de aplicación.

Además de los bloqueos de exclusividad, un proceso de aplicación que se ejecute al nivel CS tiene, como mínimo, un bloqueo de compartimiento para la fila actual de cada cursor.

Lectura no confirmada (UR)

Para una operación `SELECT INTO`, una operación `FETCH` con un cursor de sólo lectura, una operación de selección completa utilizada en `INSERT`, una operación de selección completa de fila en `UPDATE` o una operación de selección completa escalar (dondequiera que se utilice), el nivel de Lectura no confirmada permite que:

- Otros procesos de aplicación cambian cualquier fila leída durante la unidad de trabajo.
- Se lea cualquier fila cambiada por otro proceso de aplicación incluso si dicho proceso de aplicación no ha confirmado el cambio.

Para otras operaciones, se aplican las reglas del nivel CS.

Comparación de los niveles de aislamiento

En el “Apéndice I. Comparación de niveles de aislamiento” en la página 1453 se puede encontrar una comparación de los cuatro niveles de aislamiento.

Consultas

Una *consulta* es un componente de determinadas sentencias de SQL que especifica una tabla resultante (temporal).

Para obtener una descripción completa de las consultas, consulte el “Capítulo 5. Consultas” en la página 471.

Expresiones de tabla

Una *expresión de tabla* crea una tabla resultante temporal a partir de una consulta simple. Las cláusulas precisan adicionalmente la tabla resultante. Por ejemplo, puede utilizar una expresión de tabla como consulta para seleccionar todos los directores de varios departamentos y especificar que deben tener más de 15 años de experiencia laboral y que tienen que estar en la sucursal de Nueva York.

Expresiones de tabla comunes

Una *expresión de tabla común* es como una vista temporal de una consulta compleja y se puede hacer referencia a ella en otros lugares de la consulta. Por ejemplo, se puede utilizar en lugar de una vista, evitando de este modo la necesidad de crear una vista. Todos los usos de una expresión de tabla común específica en una consulta compleja comparten la misma vista temporal.

Se puede utilizar de forma repetida una expresión de tabla común en una consulta para dar soporte a aplicaciones como, por ejemplo, sistemas de reservas de líneas aéreas, generadores de listas de materiales (BOM) y planificación de redes. El "Apéndice M. Ejemplo de recurrencia: Lista de material" en la página 1499 contiene un conjunto de ejemplos de una aplicación BOM.

Procesos, simultaneidad y recuperación de aplicaciones

Todos los programas SQL se ejecutan como parte de un *proceso de aplicación* o agente. Un proceso de aplicación implica la ejecución de uno o varios programas y es la unidad a la que el gestor de bases de datos asigna los distintos recursos y bloqueos. Los distintos procesos de la aplicación pueden implicar la ejecución de programas diferentes, o distintas ejecuciones del mismo programa.

Puede que más de un proceso de aplicación solicite acceso a los mismos datos al mismo tiempo. El *bloqueo* es el mecanismo que se utiliza para mantener la integridad de los datos en tales condiciones, con lo que se evita, por ejemplo, que dos procesos de la aplicación actualicen simultáneamente la misma fila de datos.

El gestor de bases de datos adquiere bloqueos a fin de evitar que los cambios no confirmados efectuados por un proceso de aplicación sean percibidos accidentalmente por otro proceso. El gestor de bases de datos libera todos los bloqueos que ha adquirido y retenido en nombre de un proceso de aplicación cuando finaliza dicho proceso. Sin embargo, un proceso de aplicación puede solicitar explícitamente que se liberen antes los bloqueos. Esto se consigue utilizando una operación de *confirmación* que libera bloqueos adquiridos durante la unidad de trabajo y también confirma cambios en la base de datos durante la unidad de trabajo.

El gestor de bases de datos proporciona un medio de *restitución* de los cambios no confirmados realizados por un proceso de aplicación. Esto podría ser necesario en caso de error en un proceso de aplicación o si se produce un *punto muerto* o un tiempo excedido por bloqueo. Sin embargo, el propio proceso de aplicación puede solicitar de modo explícito que se restituyan los cambios en la base de datos. Esta operación se denomina *retrotracción*.

Una *unidad de trabajo* es una secuencia recuperable de operaciones dentro de un proceso de aplicación. Una unidad de trabajo comienza al iniciar un proceso de aplicación. También se inicia una unidad de trabajo cuando la anterior acaba por otro proceso que no sea la finalización del proceso de aplicación. Una unidad de trabajo finaliza mediante una operación de confirmación, de retrotracción o por el final de un proceso de aplicación. La operación de retrotracción o confirmación sólo afecta a los cambios realizados en la base de datos durante la unidad de trabajo que finaliza.

Mientras estos cambios permanecen sin confirmar, otros procesos de aplicaciones no pueden percibirlos y dichos cambios pueden restituirse. Esto es cierto excepto cuando se utiliza la lectura no confirmada (UR) de nivel de aislamiento, como se describe en el apartado “Lectura no confirmada (UR)” en la página 26. Una vez confirmados, los demás procesos de aplicación pueden acceder a estos cambios de la base de datos y éstos ya no se pueden restituir mediante una retrotracción.

La CLI de DB2 y el SQL incorporado permiten una modalidad de conexión llamada *transacciones concurrentes* que soporta múltiples conexiones, cada una de las cuales es una transacción independiente. Una aplicación puede tener múltiples conexiones concurrentes con la misma base de datos. Consulte la publicación *Application Development Guide* para obtener detalles sobre el acceso a bases de datos de múltiples hebras.

Los bloqueos adquiridos por el gestor de bases de datos para un proceso de aplicación se conservan hasta el final de una unidad de trabajo. Son excepciones a esta regla el nivel de aislamiento de estabilidad del cursor, donde el bloqueo se libera cuando el cursor se desplaza de una fila a otra, y el nivel de lectura no confirmada, en el que no se obtienen bloqueos (vea “Nivel de aislamiento” en la página 23).

El inicio y la finalización de una unidad de trabajo define los puntos de coherencia en un proceso de aplicación. Por ejemplo, una transacción bancaria puede incluir la transferencia de fondos de una cuenta a otra.

Una transacción de este tipo necesitaría que dichos fondos se restaran de la primera cuenta y se sumaran a la segunda.

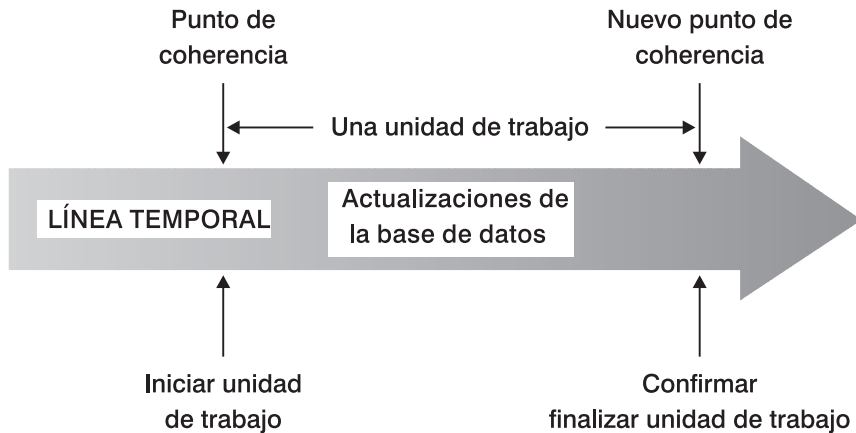


Figura 2. Unidad de trabajo con una sentencia de confirmación

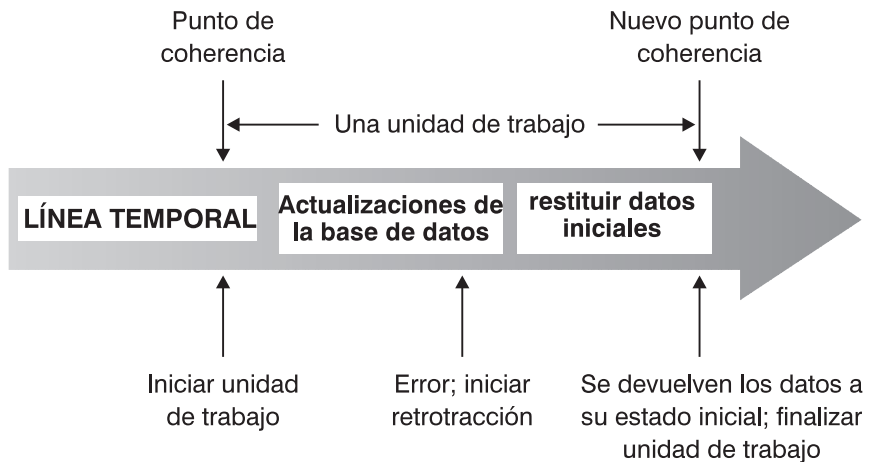


Figura 3. Unidad de trabajo con una sentencia de retroacción

Después de esta sustracción, los datos son incoherentes. La coherencia sólo queda restablecida cuando los fondos se han sumado a la segunda cuenta. Cuando se hayan completado los dos pasos, podrá utilizarse la operación de confirmación para finalizar la unidad de trabajo, con lo que los cambios estarán disponibles para otros procesos de aplicación.

Si se produce una anomalía antes de que finalice la unidad de trabajo, el gestor de bases de datos retrotraerá los cambios no confirmados para restablecer la coherencia de los datos que se presupone que existía al iniciar la unidad de trabajo.

Nota: No se puede impedir nunca que un proceso de aplicación realice operaciones debido a sus propios bloqueos. Sin embargo, si una aplicación utiliza transacciones concurrentes, los bloqueos de una transacción pueden afectar la operación de una transacción concurrente. Consulte el manual *Application Development Guide* para obtener detalles.

SQL interactivo

Las sentencias de SQL *interactivas* las entra un usuario mediante una interfaz como el procesador de línea de mandatos (CLP) o el Centro de mandatos. Estas sentencias se procesan como sentencias de SQL dinámicas. Por ejemplo, una sentencia SELECT interactiva puede procesarse dinámicamente utilizando las sentencias DECLARE CURSOR, PREPARE, DESCRIBE, OPEN, FETCH y CLOSE.

La publicación *Consulta de mandatos* lista los mandatos que se pueden emitir utilizando el CLP o recursos y productos similares.

SQL incorporado

Las sentencias de SQL incorporado son sentencias de SQL escritas en lenguajes de programación de aplicaciones como, por ejemplo, C y Java. Antes de que se compile el programa de aplicación, un precompilador SQL preprocesa estas sentencias. Existen dos tipos de SQL incorporado: estático y dinámico.

SQL estático

El formato fuente de una sentencia de SQL estática está incluido dentro un programa de aplicación escrito en un lenguaje principal como, por ejemplo, el COBOL. La sentencia se prepara antes de la ejecución del programa y el formato operativo de la sentencia persiste después de la ejecución del programa.

Un precompilador SQL debe procesar los programas fuente que contienen sentencias de SQL estáticas antes de que sean compilados. El precompilador convierte las sentencias de SQL en comentarios del lenguaje principal y genera sentencias del lenguaje principal para invocar el gestor de bases de datos. La sintaxis de las sentencias de SQL se comprueba durante el proceso de precompilación.

La preparación de un programa de aplicación SQL incluye la precompilación, el enlace de las sentencias de SQL estáticas con la base de datos de destino y la compilación del programa fuente modificado. Los pasos se especifican en el manual *Application Development Guide*.

SQL dinámico

Los programas que contienen sentencias de SQL dinámicas incorporadas deben precompilarse igual que los que contienen sentencias de SQL estáticas pero, a diferencia del SQL estático, las sentencias de SQL dinámicas se construyen y preparan durante en tiempo de ejecución. El texto de la sentencia de SQL se prepara y ejecuta utilizando las sentencias PREPARE y EXECUTE o bien la sentencia EXECUTE IMMEDIATE. Si se trata de una sentencia SELECT, también se puede ejecutar con operaciones del cursor.

Para obtener más información sobre cómo procesar cursores en sentencias de SQL dinámicas, consulte la publicación *Application Development Guide*.

Interfaz de nivel de llamada (CLI) de DB2 y Open Database Connectivity (ODBC)

La interfaz de nivel de llamada de DB2 es una interfaz de programación de aplicaciones en la que se proporcionan funciones a los programas de aplicación para que procesen las sentencias de SQL dinámico. Los programas CLI también se pueden compilar mediante un SDK (Software Developer's Kit) de ODBC (Open Database Connectivity), suministrado por Microsoft u otro proveedor, que permite acceder a fuentes de datos ODBC. A diferencia de lo que sucede con el SQL incorporado, no se necesita realizar ninguna precompilación. Las aplicaciones desarrolladas utilizando esta interfaz pueden ejecutarse en diversas bases de datos sin compilarse en cada una de ellas. A través de esta interfaz, las aplicaciones utilizan llamadas a procedimientos en tiempo de ejecución para conectarse a bases de datos, para emitir sentencias de SQL y para obtener datos e información de estado.

La interfaz CLI de DB2 proporciona muchas características que no están disponibles en el SQL incorporado. Algunas de estas características son:

- La CLI proporciona llamadas de función que soportan un modo coherente de consultar y recuperar la información de catálogo del sistema de bases de datos a través de la familia de sistemas de gestión de bases de datos DB2. Esto reduce la necesidad de escribir consultas de catálogo específicas de los servidores de bases de datos.
- CLI proporciona la capacidad de desplazarse con un cursor de estas maneras:
 - Hacia adelante, una o más filas
 - Hacia atrás, una o más filas
 - Hacia adelante desde la primera fila, una o más filas
 - Hacia atrás desde la última fila, una o más filas
 - Desde una posición en el cursor almacenada previamente
- Los procedimientos almacenados llamados desde programas de aplicación que se han escrito con la CLI pueden devolver conjuntos resultantes a esos programas.

El manual *CLI Guide and Reference* describe las API a las que se da soporte con esta interfaz.

Programas JDBC (Java Database Connectivity) y SQLJ (SQL incorporado para Java)

DB2 Universal Database implanta dos API de programación Java basadas en estándares: Conectividad de bases de datos Java (JDBC) y SQL incorporado para Java (SQLJ). Se pueden utilizar las dos para crear aplicaciones Java y applets que acceden a DB2.

Las llamadas JDBC se convierten en llamadas CLI de DB2 a través de métodos nativos Java. Las peticiones JDBC fluyen del DB2 cliente a través de la CLI de DB2 hasta el servidor de DB2. JDBC no puede utilizar SQL estático.

Las aplicaciones SQLJ utilizan JDBC como base para tareas como conectarse a bases de datos y manejar errores de SQL, pero también pueden contener sentencias de SQL estáticas intercaladas en los archivos fuente SQLJ. Un archivo fuente SQLJ se ha de convertir con el conversor SQLJ para que se pueda compilar el código Java resultante.

Para obtener más información sobre las aplicaciones JDBC y SQLJ, consulte la publicación *Application Development Guide*.

Paquetes

Un *paquete* es un objeto que contiene todas las *secciones* de un archivo fuente individual. Una sección es el formato compilado de una sentencia de SQL. Mientras que cada una de las secciones corresponde a una sentencia, cada sentencia no tiene necesariamente una sección. Las secciones creadas para el SQL estático son comparables al formato enlazado u operativo de las sentencias de SQL. Las secciones creadas para el SQL dinámico son comparables a las estructuras de control de espacios reservados utilizadas en tiempo de ejecución.

Los paquetes se generan durante la preparación del programa. Consulte el manual *Application Development Guide* para obtener más información sobre paquetes.

Vistas de catálogo

El gestor de bases de datos mantiene un conjunto de vistas y tablas base que contienen información sobre los datos que se encuentran bajo su control. Estas vistas y las tablas base se conocen en su conjunto como el *catálogo*. Contienen información sobre los objetos de la base de datos, por ejemplo tablas, vistas, índices, paquetes y funciones.

Las vistas de catálogo son como las demás vistas de base de datos. Se pueden utilizar las sentencias de SQL para consultar los datos de las vistas de catálogo de la misma manera en que se recuperan los datos de cualquier otra vista del sistema. El gestor de bases de datos garantiza que el catálogo contenga, en todo momento, una descripción exacta de los objetos de la base de datos. Para modificar ciertos valores del catálogo puede utilizarse un conjunto de vistas actualizables del catálogo (consulte el apartado “Vistas de catálogo actualizables” en la página 1282).

En el catálogo también se incluye información estadística. La información estadística la actualiza un administrador mediante la ejecución de programas de utilidad o a través de sentencias de actualización por parte de usuarios debidamente autorizados.

Las vistas de catálogo aparecen listadas en el “Apéndice D. Vistas de catálogo” en la página 1281.

Conversión de caracteres

Una *serie* es una secuencia de bytes que puede representar caracteres. En una serie, todos los caracteres se representan mediante una representación común de códigos. En algunos casos, puede ser necesario convertir estos caracteres a una representación de códigos diferente. El proceso de conversión se conoce como *conversión de caracteres*.

La conversión de caracteres, cuando es necesaria, es automática y es transparente para la aplicación cuando es satisfactoria. Por lo tanto, no es necesario tener conocimientos acerca de la conversión cuando todas las series implicadas en la ejecución de una sentencia se representan del mismo modo. Con frecuencia, éste es el caso de las instalaciones *autónomas* y de las redes dentro del mismo país. Por lo tanto, para muchos lectores, la conversión de caracteres puede carecer de importancia.

La conversión de caracteres puede producirse cuando una sentencia de SQL se ejecuta de manera remota. Por ejemplo, considere estos dos casos:

- Los valores de las variables del lenguaje principal enviadas desde el petionario de aplicaciones al servidor de aplicaciones.
- Los valores de las columnas del resultado enviados desde el servidor de aplicaciones al petionario de aplicaciones.

En cualquiera de los casos, la representación de serie puede ser diferente en el sistema emisor y el sistema receptor. La conversión también puede darse durante las operaciones de series en el mismo sistema.

La siguiente lista define algunos de los términos utilizados al explicar la conversión de caracteres.

juego de caracteres

Juego definido de caracteres. Por ejemplo, el siguiente juego de caracteres aparece en varias páginas de códigos:

- 26 letras no acentuadas de la A a la Z
- 26 letras no acentuadas de la a a la z
- dígitos del 0 al 9
- . , ; ? () ' " / - _ & + % * = < >

página de códigos

Conjunto de asignaciones de caracteres a elementos de código. En el esquema de codificación ASCII para la página de códigos 850, por ejemplo, se asigna a la "A" el elemento de código X'41' y a la "B" el elemento de código X'42'. En una página de códigos, cada elemento de código tiene un solo significado específico. Una página de códigos es un atributo de la base de datos. Cuando un programa de aplicación se conecta a la base de datos, el gestor de bases de datos determina la página de códigos de la aplicación.

elemento de código

Patrón de bits que representa de forma exclusiva un carácter.

esquema de codificación

Conjunto de reglas utilizadas para representar datos de tipo carácter, por ejemplo:

- ASCII de un solo byte
- EBCDIC de un solo byte
- ASCII de doble byte
- ASCII mixto de un solo byte y de doble byte

Juegos de caracteres y páginas de códigos

El siguiente ejemplo indica cómo un juego de caracteres típico puede correlacionarse con diferentes elementos de código de dos páginas de códigos distintas.

Página de códigos: pp1 (ASCII)

	0	1	2	3	4	5		E	F
0				0	@	P		Â	
1				1	A	Q		À	α
2			"	2	B	R		Å	β
3				3	C	S		Á	γ
4				4	D	T		Ã	δ
5			%	5	E	U		Ä	ε
E			.	>	N			5/8	Ö
F			/	*	0			®	

Punto de código: 2F Juego de caracteres ss1 (en página de códigos pp1)

Página de códigos: pp2 (EBCDIC)

	0	1		A	B	C	D	E	F
0					#				0
1					\$	A	J		1
2				s	%	B	K	S	2
3				t	¬	C	L	T	3
4				u	*	D	M	U	4
5				v	(E	N	V	5
E					!	:	Â	}	
F				À	ç	;	Á	{	

Juego de caracteres ss1 (en página de códigos pp2)

Figura 4. Correlación de un juego de caracteres en diferentes páginas de códigos

Incluso con el mismo esquema de codificación, existen muchas páginas de códigos diferentes y el mismo elemento de código puede representar un carácter diferente en páginas de código diferentes. Además, un byte de una serie de caracteres no representa necesariamente un carácter de un juego de caracteres de un solo byte (SBCS). Las series de caracteres también se utilizan para datos de bits y mixtos. Los *datos mixtos* son una combinación de caracteres de un solo byte, de doble byte o de múltiples bytes. Los *datos de bits* (columnas definidas como FOR BIT DATA o BLOB, o series binarias) no están asociados a ningún juego de caracteres.

Atributos de páginas de códigos

El gestor de bases de datos determina los atributos de páginas de códigos para todas las series de caracteres cuando una aplicación se enlaza con una base de datos. Los atributos de página de códigos posibles son:

Página de códigos de base de datos

Página de códigos de base de datos almacenada en los archivos de configuración de la base de datos. Este valor de página de códigos se determina cuando se crea la base de datos y no puede modificarse.

Página de códigos de aplicación

Página de códigos bajo la que se ejecuta la aplicación. Recuerde que ésta no es necesariamente la misma página de códigos bajo la que se ha enlazado la aplicación. (Consulte la publicación *Application Development Guide* para obtener información adicional sobre cómo enlazar y ejecutar programas de aplicación.)

Página de códigos 0

Ésta representa una serie derivada de una expresión que contiene un valor FOR BIT DATA o BLOB.

Atributos de página de códigos de series

Las páginas de códigos de series de caracteres pueden tener los atributos siguientes:

- Las columnas pueden estar en la página de códigos de la base de datos o en la página de códigos 0 (si se han definido como de caracteres FOR BIT DATA o BLOB).
- Las constantes y los registros especiales (por ejemplo, USER, CURRENT SERVER) están en la página de códigos de la base de datos. Recuerde que las constantes se convierten a la página de códigos de la base de datos cuando una sentencia de SQL se enlaza con la base de datos.
- Las variables de lenguaje principal de entrada se encuentran en la página de códigos de la aplicación.

Se utiliza un conjunto de reglas para determinar los atributos de página de códigos para las operaciones que combinan objetos de serie como, por ejemplo, los resultados de operaciones escalares, concatenaciones u operaciones de conjuntos. Durante el tiempo de ejecución, los atributos de la página de códigos se utilizan para determinar todos los requisitos para las conversiones de páginas de códigos de las series.

Para obtener más detalles sobre la conversión de caracteres, consulte los apartados :

“Reglas de conversión para asignaciones de series” en la página 108, que explica las reglas sobre asignaciones de series

“Reglas para las conversiones de series” en la página 124, que explica las reglas sobre las conversiones cuando se comparan o se combinan series de caracteres.

Supervisores de sucesos

Un *supervisor de sucesos* realiza el seguimiento de datos específicos como resultado de un suceso. Por ejemplo, iniciar la base de datos podría ser un suceso que obligase al supervisor de sucesos a realizar un seguimiento del número de usuarios del sistema, tomando cada hora una instantánea de los ID de autorización que utilizan la base de datos.

Los supervisores de sucesos se activan o desactivan con una sentencia (SET EVENT MONITOR STATE). La función EVENT_MON_STATE puede utilizarse para buscar el estado actual (activo o inactivo) de un supervisor de sucesos.

Desencadenantes

Un *desencadenante* define un conjunto de acciones que se ejecutan en, o se activan por, una operación de supresión, inserción o actualización en una tabla especificada. Cuando se ejecuta una de estas operaciones SQL, se dice que el desencadenante está *activado*.

Los desencadenantes pueden utilizarse junto con las restricciones de referencia y de comprobación para imponer las reglas de integridad de los datos. Los desencadenantes también pueden utilizarse para provocar actualizaciones en otras tablas, para transformar o generar valores automáticamente en las filas insertadas o actualizadas, o para invocar funciones que realicen tareas como la de emitir alertas.

Los desencadenantes son un mecanismo útil para definir e imponer reglas empresariales *transicionales*, que son reglas que incluyen diferentes estados de los datos (por ejemplo, el salario no se puede aumentar más del 10 por ciento). En las reglas que no afectan a más de un estado de los datos, hay que tener en cuenta las restricciones de integridad de referencia y de comprobación.

La utilización de desencadenantes proporciona la lógica para imponer las reglas empresariales en la base de datos. Esto significa que las aplicaciones que utilizan las tablas no son responsables de imponer dichas reglas. La lógica impuesta de forma centralizada a todas las tablas significa un mantenimiento más sencillo, ya que no es necesario cambiar el programa de aplicación cuando cambia la lógica.

Los desencadenantes son opcionales y se definen mediante la sentencia CREATE TRIGGER.

Al crear un desencadenante se definen diversos criterios. Éstos se utilizan para determinar cuándo debe activarse un desencadenante.

- La *tabla sujeto* define la tabla para la que se define el desencadenante.

- El *suceso desencadenante* define una operación SQL específica que modifica la tabla sujeto. La operación puede ser la supresión, inserción o actualización.
- El *momento de activación del desencadenante* define si el desencadenante debe activarse antes o después de haberse producido el suceso desencadenante en la tabla sujeto.

La sentencia que hace que se active un desencadenante incluye un *conjunto de filas afectadas*. Éstas son las filas de la tabla sujeto que se suprimen, insertan o actualizan. La *granularidad del desencadenante* define si las acciones del desencadenante se realizan una vez para la sentencia o una vez para cada una de las filas del conjunto de filas afectadas.

La *acción activada* consta de una condición de búsqueda opcional y un conjunto de sentencias de SQL que se ejecutan siempre que se activa el desencadenante. Las sentencias de SQL sólo se ejecutan si la condición de búsqueda se evalúa como verdadera. Cuando la hora de activación del desencadenante es anterior al suceso desencadenante, las acciones desencadenadas pueden incluir sentencias que seleccionan, establecen variables de transición y señalan estados de SQL. Cuando la hora de activación del desencadenante es posterior al suceso desencadenante, las acciones desencadenadas pueden incluir sentencias que seleccionan, actualizan, insertan, suprimen y señalan estados de SQL.

La acción desencadenada puede hacer referencia a los valores del conjunto de filas afectadas utilizando *variables de transición*. Las variables de transición utilizan los nombres de las columnas de la tabla sujeto calificados por un nombre especificado que identifica si la referencia es al valor anterior (previo a la actualización) o al valor nuevo (posterior a la actualización). El nuevo valor también se puede cambiar utilizando la sentencia de variable de transición SET antes de actualizar o insertar desencadenantes. Otro método para hacer referencia a los valores del conjunto de filas afectadas consiste en utilizar *tablas de transición*. Las tablas de transición también utilizan los nombres de las columnas de la tabla sujeto pero especifican un nombre para permitir que el conjunto completo de filas afectadas se trate como una tabla. Las tablas de transición sólo se pueden utilizar en desencadenantes posteriores; y las tablas de transición se pueden definir para valores antiguos y nuevos.

Se pueden especificar varios desencadenantes para una combinación de tabla, suceso o momento de activación. El orden en el que se activan los desencadenantes es el mismo que el orden en que se han creado. Por consiguiente, el desencadenante creado más recientemente es el último desencadenante que se activa.

La activación de un desencadenante puede producir una *cascada de desencadenantes*. Es el resultado de la activación de un desencadenante que

ejecuta sentencias de SQL que provocan la activación de otros desencadenantes o incluso del mismo otra vez. Las acciones desencadenadas también pueden producir actualizaciones como resultado de la modificación original o como resultado de las reglas de integridad de referencia para las supresiones que pueden producir la activación de desencadenantes adicionales. Con una cascada de desencadenantes, se puede activar una cadena de desencadenantes y reglas de supresión de integridad de referencia, lo que puede producir un cambio significativo en la base de datos como resultado de una sola sentencia de supresión, inserción o actualización.

Espacios de tablas y otras estructuras de almacenamiento

Las estructuras de almacenamiento contienen los objetos de la base de datos. Las estructuras básicas de almacenamiento gestionadas por el gestor de bases de datos son los espacios de tablas. Un *espacio de tablas* es una estructura de almacenamiento que contiene tablas, índices, objetos grandes y datos definidos con un tipo de datos LONG. Existen dos tipos de espacios de tablas:

Espacio de tablas con espacio gestionado por la base de datos (espacio de tablas DMS)

Espacio de tablas cuyo espacio lo gestiona el gestor de bases de datos.

Espacio de tablas con espacio gestionado por el sistema (espacio de tablas SMS) Espacio de tablas cuyo espacio lo gestiona el sistema operativo.

Todos los espacios de tablas constan de contenedores. Un *contenedor* describe dónde se almacenan los objetos, por ejemplo, algunas tablas. Por ejemplo, un subdirectorio de un sistema de archivos puede ser un contenedor.

Para obtener más información sobre espacios de tablas y contenedores, consulte el apartado “CREATE TABLESPACE” en la página 893 o el manual *Administration Guide*.

Los datos que se leen en los contenedores de espacios de tablas se colocan en un área de la memoria denominada *agrupación de almacenamientos intermedios*. Una agrupación de almacenamientos intermedios está asociada a un espacio de tablas que permite controlar qué datos comparten las mismas áreas de memoria para el almacenamiento intermedio de datos. Para obtener más información sobre las agrupaciones de almacenamientos intermedios, consulte el apartado “CREATE BUFFERPOOL” en la página 673 o la publicación *Administration Guide*.

Una base de datos particionada permite que los datos se repartan entre distintas particiones de la base de datos. Las particiones incluidas se determinan por el *grupo de nodos* asignado al espacio de tablas. Un grupo de nodos es un grupo de una o varias particiones que se definen como parte de la base de datos. Un espacio de tablas incluye uno o varios contenedores para

cada partición del grupo de nodos. Se asocia un *mapa de particionamiento* a cada grupo de nodos. El gestor de bases de datos utiliza el mapa de particionamiento para determinar la partición del grupo de nodos que se almacenará en una fila de datos determinada. Para obtener más información sobre los grupos de nodos y el particionamiento de datos, consulte el apartado “Partición de datos entre múltiples particiones”, el apartado “CREATE NODEGROUP” en la página 801 o la publicación *Administration Guide*.

Una tabla también puede incluir columnas que registren enlaces con datos almacenados en archivos externos. El mecanismo para ello es el tipo de datos DATALINK. Un valor DATALINK que se registre en una tabla regular apunta a un archivo almacenado en un servidor de archivos externos.

El DB2 Data Links Manager, que se instala en un servidor de archivos, funciona junto con DB2 para proporcionar la siguiente funcionalidad opcional:

- Integridad de referencia para asegurar que los archivos enlazados actualmente a DB2 no se suprimen ni se redennominan.
- Seguridad para asegurar que sólo las personas que tengan privilegios de SQL adecuados en la columna DATALINK pueden leer los archivos enlazados a dicha columna.
- Coordinación en la copia de seguridad y recuperación del archivo.

El producto DB2 Data Links Manager comprende los siguientes recursos:

Data Links File Manager

Registra todos los archivos de un servidor de archivos en particular que estén enlazados con DB2.

Data Links Filesystem Filter

Filtra mandatos de sistema de archivos para garantizar que los archivos registrados no se supriman ni se redennominen. Opcionalmente, también filtra mandatos para garantizar que exista una autorización de acceso correspondiente.

Para obtener más información sobre DB2 Data Links Manager, consulte la publicación *Administration Guide*.

Partición de datos entre múltiples particiones

DB2 permite una gran flexibilidad para repartir los datos entre múltiples particiones (nodos) de una base de datos particionada. Los usuarios pueden elegir la forma de particionar los datos mediante la declaración de claves de particionamiento y pueden determinar qué particiones y en cuántas de ellas pueden repartirse los datos de tabla mediante la selección del grupo de nodos y el espacio de tablas en el que se deben almacenar los datos. Además, un

mapa de particionamiento (que puede actualizar el usuario) especifica la correlación de los valores de clave de particionamiento para las particiones. Esto posibilita la paralelización flexible de la carga de trabajo para tablas grandes, mientras que permite que se almacenen las tablas mas pequeñas en una o en un pequeño número de particiones si lo elige el diseñador de la aplicación. Cada partición local puede tener índices locales acerca de los datos que almacenan para proporcionar un acceso de datos local de alto rendimiento.

Una base de datos particionada da soporte a un modelo de almacenamiento particionado en el que la clave de particionamiento se utiliza para particionar los datos de tabla en un conjunto de particiones de la base de datos. Los datos de índice también se particionan con sus tablas correspondientes y se almacenan localmente en cada partición.

Antes de que se puedan utilizar las particiones para almacenar datos de la base de datos, deben definirse en el gestor de bases de datos. Las particiones se definen en un archivo llamado `db2nodes.cfg`. Consulte el manual *Administration Guide* para obtener más detalles acerca de la definición de particiones.

La clave de particionamiento para una tabla de un espacio de tablas de un grupo de nodos particionado se especifica en la sentencia CREATE TABLE (o en la sentencia ALTER TABLE). Si no se especifica, se crea por omisión una clave de particionamiento para una tabla a partir de la primera columna de la clave primaria. Si no se especifica ninguna clave primaria, la clave de particionamiento por omisión es la primera columna definida en la tabla que tiene un tipo de datos que no es LONG ni LOB. Las tablas particionadas deben tener, como mínimo, una columna que no tenga el tipo de datos LONG ni LOB. Una tabla de un espacio de tablas de un grupo de nodos de una sola partición sólo tendrá una clave de particionamiento si se especifica explícitamente.

El *particionamiento por cálculo de clave* se utiliza para colocar una fila en una partición, de la manera siguiente.

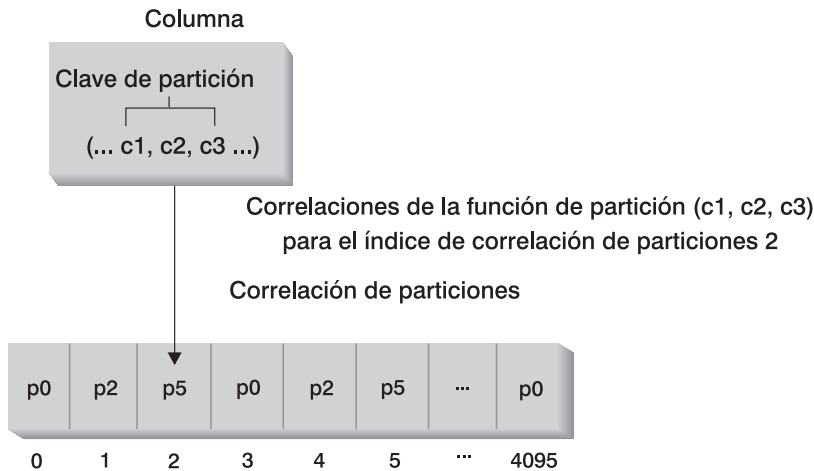
1. Un algoritmo de cálculo de claves (función de particionamiento) se aplica a todas las columnas de la clave de particionamiento, lo que da como resultado la generación de un índice de mapa de particionamiento.
2. Este índice de mapa de particionamiento se utiliza como índice para el mapa de particionamiento. El número de partición correspondiente a ese índice en el mapa de particionamiento es la partición donde se almacena la fila.
3. Los mapas de particionamiento se asocian a grupos de nodos y se crean tablas en los espacios de tablas que están en grupos de nodos.

DB2 da soporte a la *desagrupación parcial*, que significa que la tabla se puede particionar en un subconjunto de particiones del sistema (es decir, un grupo de nodos). Las tablas no se tienen que particionar en todas las particiones del sistema.

Mapas de particionamiento

Cada grupo de nodos está asociado con un *mapa de particionamiento*, que es una matriz de 4.096 números de partición. Para determinar la partición en la que se almacena una fila se utiliza, como índice en el mapa de particionamiento, el índice de mapa de particionamiento producido por la función de particionamiento para cada fila de una tabla.

La Figura 5 muestra cómo la fila del valor de clave particionada (c1, c2, c3) se correlaciona con el índice 2 del mapa de particionamiento que, a su vez, hace referencia a la partición p5.



Las particiones del grupo de nodos son p0, p2 y p5

Nota: Los números de partición empiezan en 0.

Figura 5. Distribución de datos

El mapa de particionamiento se puede cambiar, permitiendo cambiar la distribución de los datos sin modificar la clave de particionamiento ni los datos reales. El nuevo mapa de particionamiento se especifica como parte del mandato REDISTRIBUTE NODEGROUP o de la interfaz de programación de aplicaciones (API) sqludrtd, que lo utilizan para redistribuir las tablas en el grupo de nodos. Consulte la publicación *Consulta de mandatos* o la publicación *Administrative API Reference* para obtener información adicional.

Colocación de tablas

DB2 tiene la posibilidad de reconocer si los datos a los que se ha accedido para una unión o subconsulta están situados en la misma partición del mismo grupo de nodos. Cuando sucede esto, DB2 puede elegir realizar el proceso de unión o de subconsulta en la partición donde se almacenan los datos, lo cual tiene normalmente ventajas de rendimiento significativas. Esta situación se denomina colocación de tablas. Para ser consideradas tablas colocadas, las tablas deben:

- Estar en el mismo grupo de nodos (que no se esté redistribuyendo⁶).
- Tener claves de particionamiento con el mismo número de columnas.
- Hacer que las columnas correspondientes de la clave de particionamiento sean compatibles con la partición (consulte el apartado “Compatibilidad entre particiones” en la página 127).
- O estar en un grupo de nodos de partición individual definido en la misma partición.

Las filas de las tablas colocadas con el mismo valor de clave de particionamiento se colocarán en la misma partición.

Base de datos relacional distribuida

Una *base de datos relacional distribuida* consta de un conjunto de tablas y otros objetos distribuidos en distintos sistemas informáticos que están conectados entre sí. Cada sistema tiene un gestor de bases de datos relacionales para manejar las tablas de su entorno. Los gestores de bases de datos se comunican y cooperan entre sí de una manera que permite a un gestor de bases de datos determinado ejecutar sentencias de SQL en otro sistema.

Las bases de datos relacionales distribuidas se crean mediante protocolos y funciones de peticionario-servidor formales. Un *peticionario de aplicaciones* da soporte al extremo correspondiente a la aplicación en una conexión. Transforma una petición de base de datos procedente de la aplicación en protocolos de comunicaciones adecuados para ser utilizados en la red de bases de datos distribuidas. Estas peticiones se reciben y procesan por un *servidor de aplicaciones* situado en el otro extremo de la conexión. Mediante un trabajo conjunto, el peticionario de aplicaciones y el servidor de aplicaciones manejan las consideraciones acerca de las comunicaciones y la ubicación con el fin de aislar la aplicación respecto a estas consideraciones para que pueda funcionar como si estuviera accediendo a una base de datos local. En la

6. Mientras se redistribuye un grupo de nodos, las tablas del grupo de nodos pueden estar utilizando mapas de particionamiento diferentes – no están colocadas.

Figura 6 se ilustra un entorno de bases de datos relacionales distribuidas sencillo.

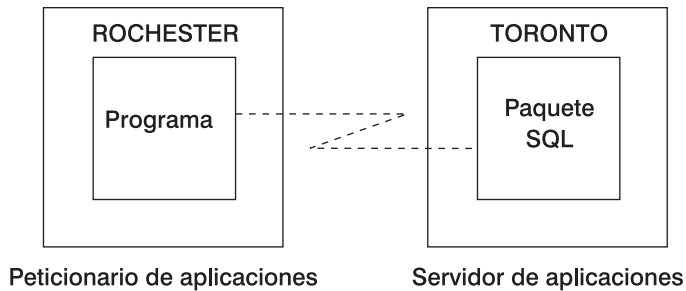


Figura 6. Entorno de bases de datos relacionales distribuidas

Para obtener más información sobre los protocolos de comunicaciones DRDA (Arquitectura de bases de datos relacionales distribuidas), consulte la publicación *Distributed Relational Database Architecture Reference* ST01-2709-00.

Servidores de aplicaciones

Un proceso de aplicación debe conectarse al servidor de aplicaciones de un gestor de bases de datos para que se puedan ejecutar las sentencias de SQL que hacen referencia a tablas o vistas. Una sentencia CONNECT establece una conexión entre un proceso de aplicación y su servidor. La CLI de DB2 y el SQL incorporado soportan una modalidad de conexión llamada *transacciones concurrentes* que permite múltiples conexiones, cada una de las cuales es una transacción independiente. Una aplicación puede tener múltiples conexiones concurrentes con la misma base de datos. Consulte la publicación *Application Development Guide* para obtener detalles sobre el acceso a bases de datos de múltiples hebras. El servidor puede cambiar cuando se ejecuta una sentencia CONNECT.

El servidor de aplicaciones puede ser local o remoto con respecto al entorno en el que se inicia el proceso. Existe un servidor de aplicaciones incluso aunque el entorno no esté utilizando bases de datos relacionales distribuidas. Este entorno incluye un directorio local que describe los servidores de aplicaciones que pueden identificarse en una sentencia CONNECT. Para obtener una descripción de los directorios locales, consulte la publicación *Administration Guide*.

Para ejecutar una sentencia de SQL estática que haga referencia a tablas o vistas, el servidor de aplicaciones utiliza la forma enlazada de la sentencia. Esta sentencia enlazada se toma de un paquete que el gestor de bases de datos ha creado previamente mediante una operación de enlace lógico.

En su mayor parte, una aplicación conectada a un servidor de aplicaciones puede utilizar las sentencias y las cláusulas soportadas por el gestor de bases de datos del servidor de aplicaciones. Esto es cierto aunque la aplicación esté ejecutándose mediante el peticionario de aplicaciones de un gestor de bases de datos que no soporta algunas esas sentencias y cláusulas.

Vea “Definiciones de servidor y opciones de servidor” en la página 59 para obtener información sobre cómo utilizar un servidor de aplicaciones para someter consultas en un sistema de fuentes de datos distribuidos.

CONNECT (Tipo 1) y CONNECT (Tipo 2)

Hay dos tipos de sentencias CONNECT:

- CONNECT (Tipo 1) da soporte a la semántica de una sola base de datos por unidad de trabajo (Unidad de trabajo remota). Vea “CONNECT (Tipo 1)” en la página 653.
- CONNECT (Tipo 2) da soporte a la semántica de varias bases de datos por unidad de trabajo (Unidad de trabajo distribuida dirigida por aplicación). Vea “CONNECT (Tipo 2)” en la página 662.

Unidad de trabajo remota

El recurso *unidad de trabajo remota* permite la preparación y ejecución remotas de las sentencias de SQL. Un proceso de aplicación del sistema A puede conectarse a un servidor de aplicaciones del sistema B y, desde una o más unidades de trabajo, ejecutar un número cualquiera de sentencias de SQL estáticas o dinámicas que hagan referencia a objetos de B. Al finalizar una unidad de trabajo en B, el proceso de aplicación puede conectarse a un servidor de aplicaciones del sistema C y así sucesivamente.

La mayoría de las sentencias de SQL pueden prepararse y ejecutarse de forma remota, con las restricciones siguientes:

- Todos los objetos a los que se hace referencia en una sola sentencia de SQL deben ser gestionados por el mismo servidor de aplicaciones
- Todas las sentencias de SQL de una unidad de trabajo deben ser ejecutadas por el mismo servidor de aplicaciones

Gestión de conexión de la unidad de trabajo remota

En este apartado se analizan los estados de conexión en los que puede entrar un proceso de aplicación.

Estados de conexión: Un proceso de aplicación se encuentra en cualquier momento en uno de cuatro estados:

- Conectable y conectado.
- No conectable y conectado.
- Conectable y no conectado.
- Conectable implícitamente (si la conexión implícita se encuentra disponible).

Si está disponible la conexión implícita (consulte la Figura 7 en la página 48), el proceso de aplicación está, inicialmente, en el estado *conectable implícitamente*. Si la conexión implícita no está disponible (consulte la Figura 8 en la página 49), el proceso de aplicación está, inicialmente, en el estado *conectable y no conectado*.

La disponibilidad de la conexión implícita viene determinada por las opciones de la instalación, las variables de entorno y los valores de autenticación. Consulte la publicación *Guía rápida de iniciación* para obtener información sobre cómo establecer la conexión implícita en la instalación y la publicación *Administration Guide* para obtener información sobre las variables de entorno y los valores de autenticación.

El estado conectable y conectado:

Un proceso de aplicación está conectado a un servidor de aplicaciones y pueden ejecutarse sentencias CONNECT.

Si la conexión implícita se encuentra disponible:

- El proceso de aplicación entra en este estado cuando una sentencia CONNECT TO o una sentencia CONNECT sin operandos se ejecuta satisfactoriamente desde el estado conectable y no conectado.
- El proceso de aplicación puede entrar también en este estado desde el estado conectable implícitamente si se emite cualquier otra sentencia de SQL que no sea CONNECT RESET, DISCONNECT, SET CONNECTION ni RELEASE.

Tanto si la conexión implícita está disponible como si no lo está, se entra en este estado cuando:

- Se ejecuta satisfactoriamente una sentencia CONNECT TO desde el estado conectable y no conectado.
- Se emite satisfactoriamente una sentencia COMMIT o ROLLBACK, o bien tiene lugar una retrotracción forzada que procede de un estado no conectable y conectado.

El estado no conectable y conectado:

Un proceso de aplicación está conectado a un servidor de aplicaciones, pero no puede ejecutarse satisfactoriamente una sentencia CONNECT TO para cambiar de servidor de aplicaciones. El proceso pasa a este estado desde el estado conectable y conectado al ejecutarse cualquier sentencia de SQL que no sea una de las siguientes: CONNECT TO, CONNECT sin operandos, CONNECT RESET, DISCONNECT, SET CONNECTION, RELEASE, COMMIT o ROLLBACK.

El estado conectable y no conectado:

Un proceso de aplicación no está conectado a un servidor de aplicaciones. La única sentencia de SQL que puede ejecutarse es CONNECT TO, de lo contrario se genera un error (SQLSTATE 08003).

Tanto si la conexión implícita está disponible como si no:

- El proceso de aplicación entra en este estado si se produce un error al emitir una sentencia CONNECT TO o si se produce un error en una unidad de trabajo que provoca la pérdida de la conexión y una retrotracción. Los errores originados porque el proceso de aplicación no está en estado conectable o porque el nombre-servidor no se encuentra en el directorio local no provoca el paso a este estado.

Si la conexión implícita no está disponible:

- Las sentencias CONNECT RESET y DISCONNECT provocan una transición a este estado.

El estado conectable implícitamente:

Si la conexión implícita se encuentra disponible, éste es el estado inicial de un proceso de aplicación. La sentencia CONNECT RESET provoca una transición a este estado. Si se emite una sentencia COMMIT (confirmación) o ROLLBACK (retrotracción) en el estado no conectable y conectado seguida de una sentencia DISCONNECT en el estado conectable y conectado, también se pasa a este estado.

Las *transiciones de estado* se muestran en los siguientes diagramas.

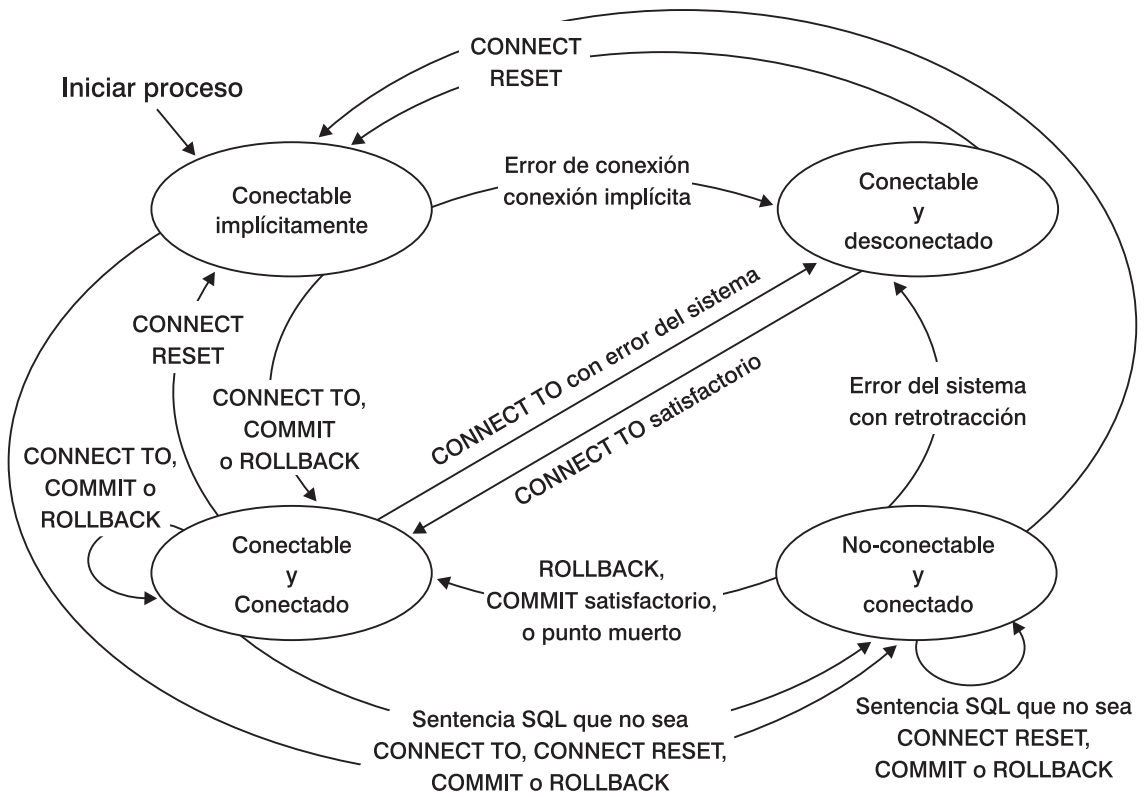


Figura 7. Transiciones de estado de conexión si la conexión implícita está disponible

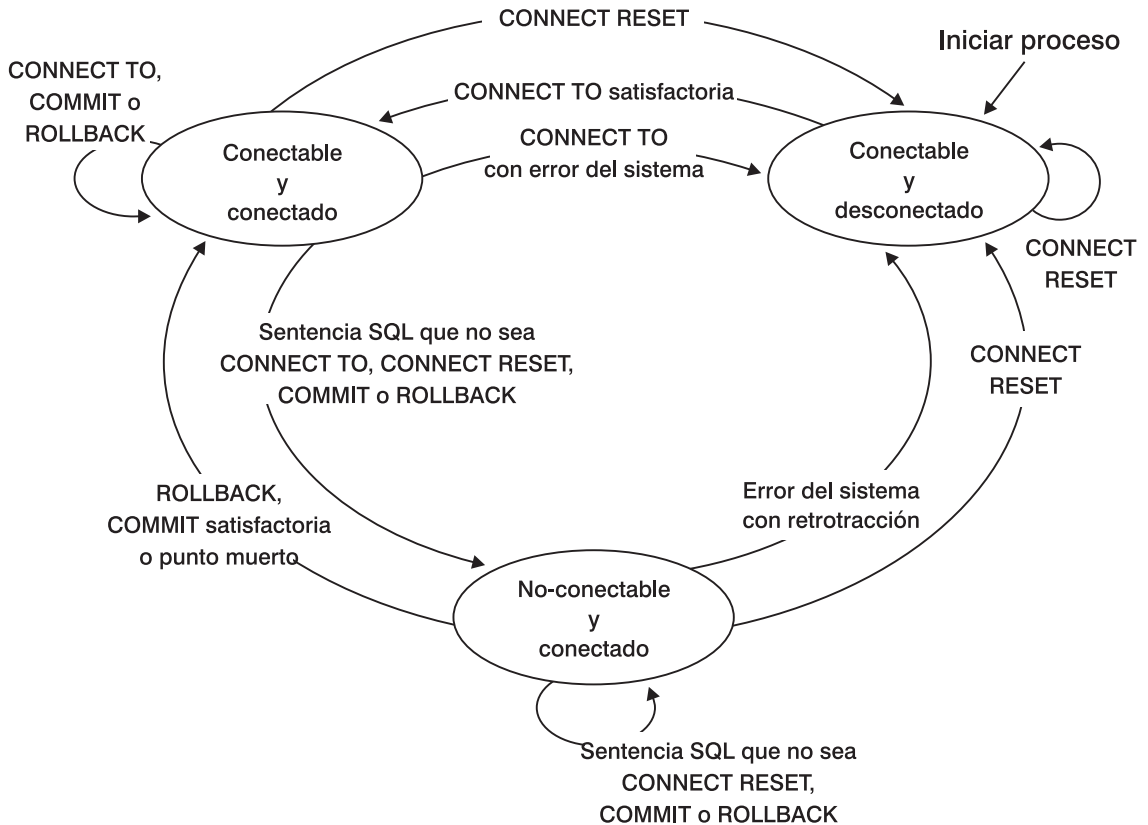


Figura 8. Transiciones de estado de conexión si la conexión implícita no está disponible

Reglas adicionales:

- No es un error ejecutar sentencias CONNECT consecutivas porque CONNECT no modifica el estado conectable del proceso de aplicación.
- Es un error ejecutar sentencias CONNECT RESET consecutivas.
- Es un error ejecutar cualquier sentencia de SQL que no sea CONNECT TO, CONNECT RESET, CONNECT sin operandos, SET CONNECTION, RELEASE, COMMIT o ROLLBACK, y luego ejecutar una sentencia CONNECT TO. Para evitar el error, se deberá ejecutar una sentencia CONNECT RESET, DISCONNECT (precedida de una sentencia COMMIT o ROLLBACK), COMMIT o ROLLBACK antes de la sentencia CONNECT TO.

Unidad de trabajo distribuida dirigida por aplicación

El recurso de unidad de trabajo distribuida dirigida por aplicación también permite la preparación y la ejecución remotas de sentencias de SQL del mismo modo que el recurso de unidad de trabajo remota. Un proceso de aplicación en el sistema A puede conectarse a un servidor de aplicaciones en el sistema B

emitiendo una sentencia CONNECT o SET CONNECTION. Después, el proceso de aplicación puede ejecutar un número cualquiera de sentencias de SQL estáticas y dinámicas que hagan referencia a objetos de B antes de finalizar la unidad de trabajo. Todos los objetos a los que se hace referencia en una sola sentencia de SQL deben ser gestionados por el mismo servidor de aplicaciones. Sin embargo, a diferencia del recurso de unidad de trabajo remota, pueden participar en la misma unidad de trabajo cualquier número de servidores de aplicaciones. Una operación de retrotracción o confirmación finaliza la unidad de trabajo.

Gestión de la conexión de la unidad de trabajo distribuida dirigida por aplicación

Una unidad de trabajo distribuida dirigida por aplicación utiliza una conexión de tipo 2. La conexión de tipo 2 conecta un proceso de aplicación al servidor de aplicaciones identificado y establece las reglas para la unidad de trabajo distribuida dirigida por aplicación.

Visión general del proceso de aplicación y de los estados de conexión

Un proceso de aplicación de tipo 2:

- Está siempre conectable.
- Está en estado *conectado* o *no conectado*.
- Tiene cero o más conexiones.

Cada conexión de un proceso de aplicación viene identificada de modo exclusivo por el seudónimo de la base de datos del servidor de aplicaciones de la conexión.

Una conexión individual tiene siempre uno de los estados de conexión siguientes:

- *actual* y *mantenido*
- *actual* y *pendiente de liberación*
- *inactivo* y *mantenido*
- *inactivo* y *pendiente de liberación*

Estados iniciales y transiciones de estado: Un proceso de aplicación de tipo 2 está inicialmente en *estado no conectado* y no tiene ninguna conexión.

Una conexión está inicialmente en estado *actual* y *mantenido*.

El siguiente diagrama muestra las transiciones de estado:

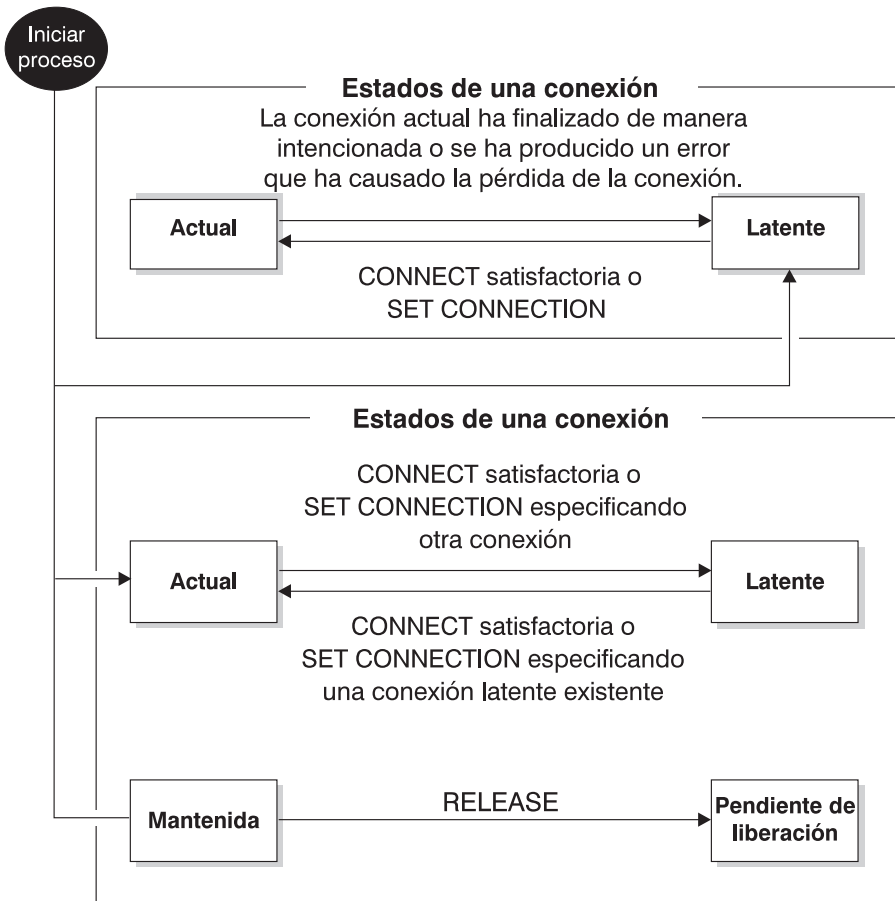


Figura 9. Conexión de la unidad de trabajo distribuida y transiciones de estado de conexión del proceso de la aplicación

Estados de conexión del proceso de aplicación: Puede establecerse un servidor de aplicaciones distinto mediante la ejecución explícita o implícita de una sentencia CONNECT. Una conexión implícita de Tipo 2 es más restrictiva que la de Tipo 1. Consulte el apartado “CONNECT (Tipo 2)” en la página 662 para obtener detalles. Se aplican las reglas siguientes a la ejecución de una sentencia CONNECT:

- Un contexto no puede tener más de una conexión al mismo servidor de aplicaciones al mismo tiempo. Consulte las publicaciones *Administration Guide* y *Application Development Guide* para obtener información sobre el soporte de múltiples conexiones con la misma DB2 Universal Database al mismo tiempo.

- Cuando un proceso de aplicación ejecuta una sentencia SET CONNECTION, el nombre de ubicación especificado debe ser una conexión existente en el conjunto de conexiones del proceso de aplicación.
- Cuando un proceso de aplicación ejecuta una sentencia CONNECT y la opción SQLRULES(STD) está en vigor, el nombre de servidor especificado no debe ser una conexión existente en el conjunto de conexiones del proceso de aplicación. Consulte el apartado “Opciones que controlan la semántica de la unidad de trabajo distribuida” en la página 54 para ver una descripción de la opción SQLRULES.

Si un proceso de aplicación tiene una conexión actual, el proceso de aplicación está en el estado *conectado*. El registro especial CURRENT SERVER contiene el nombre del servidor de aplicaciones de la conexión actual. El proceso de aplicación puede ejecutar sentencias de SQL que hagan referencia a objetos gestionados por el servidor de aplicaciones.

Un proceso de aplicación en estado no conectado pasa al estado conectado cuando ejecuta satisfactoriamente una sentencia CONNECT o SET CONNECTION. Si no existe ninguna conexión en la aplicación pero se emiten sentencias de SQL, se realizará una conexión implícita siempre que se haya definido la variable de entorno DB2DBDFT con una base de datos por omisión.

Si un proceso de aplicación no tiene una conexión actual, el proceso de aplicación está en estado *no conectado*. Las únicas sentencias de SQL que se pueden ejecutar son CONNECT, DISCONNECT ALL, DISCONNECT especificando una base de datos, SET CONNECTION, RELEASE, COMMIT y ROLLBACK.

Un proceso de aplicación en el *estado conectado* entra en el *estado no conectado* cuando finaliza de manera intencionada su conexión actual o cuando la ejecución de una sentencia de SQL no es satisfactoria debido a una anomalía que origina una operación de retrotracción en el servidor de aplicaciones y la pérdida de la conexión. Las conexiones finalizan intencionadamente al ejecutarse satisfactoriamente una sentencia DISCONNECT o al ejecutarse satisfactoriamente una operación de confirmación cuando la conexión está en *estado pendiente de liberación*.

Las distintas opciones especificadas en la opción DISCONNECT del precompilador afectan a la finalización de manera intencionada de una conexión. Si se establece en AUTOMATIC, se finalizarán todas las conexiones. Si se establece en CONDITIONAL, finalizan todas las conexiones que no tienen cursores WITH HOLD abiertos.

Estados de una conexión: Si un proceso de aplicación ejecuta una sentencia CONNECT y el peticionario de aplicaciones conoce el nombre de servidor y no está en el conjunto de conexiones existentes del proceso de aplicación, entonces:

- la conexión actual se coloca en el *estado inactivo*,
- el nombre del servidor se añade al conjunto de conexiones y
- la nueva conexión se coloca en el *estado actual* y en el *estado mantenido*.

Si el nombre de servidor ya se encuentra en el conjunto de conexiones existentes del proceso de aplicación y ésta se precompila con la opción SQLRULES(STD), se produce un error (SQLSTATE 08002).

- **Estados mantenido y pendiente de liberación:** La sentencia RELEASE controla si una conexión está en el estado mantenido o en el estado pendiente de liberación. Un *estado pendiente de liberación* significa que se ha de producir una desconexión de la conexión en la siguiente operación satisfactoria de confirmación (la retrotracción no produce ningún efecto en las conexiones). Un *estado mantenido* significa que la conexión no se ha de desconectar en la siguiente operación. Todas las conexiones se encuentran inicialmente en el estado mantenido y pueden pasar al estado pendiente de liberación utilizando la sentencia RELEASE. Una vez en el estado pendiente de liberación, una conexión no puede volver a pasar al estado mantenido. Una conexión permanecerá en el estado pendiente de liberación más allá de los límites de la unidad de trabajo si se emite una sentencia ROLLBACK o si una operación de confirmación no satisfactoria da como resultado una operación de retrotracción.

Aunque una conexión no se haya marcado explícitamente para su liberación, todavía puede desconectarse mediante una operación de confirmación, siempre que esta operación cumpla con las condiciones de la opción DISCONNECT del precompilador.

- **Estados actual e inactivo:** Sin tener en cuenta si una conexión está en el *estado mantenido* o en el *estado pendiente de liberación*, una conexión también puede estar en el *estado actual* o en el *estado inactivo*. Un *estado actual* significa que la conexión es la que se utiliza para las sentencias de SQL que se ejecutan mientras están en dicho estado. Un *estado inactivo* significa que la conexión no es actual.

Las únicas sentencias de SQL que pueden circular en una conexión inactiva son COMMIT y ROLLBACK; o DISCONNECT y RELEASE, que pueden especificar ALL (todas las conexiones) o el nombre de una base de datos específica. Las sentencias SET CONNECTION y CONNECT cambian la conexión para el servidor nombrado al *estado actual* mientras que cualquier conexión existente se coloca o permanece en el *estado inactivo*. En todo momento, sólo puede haber una conexión en el *estado actual*. Cuando una conexión inactiva pasa a ser actual en la misma unidad de trabajo, el estado

de todos los bloqueos, cursores y sentencias preparadas seguirá siendo el mismo y reflejarán la que fue su última utilización cuando la conexión era actual.

Cuando finaliza una conexión: Cuando una conexión finaliza, se desasignan todos los recursos que el proceso de aplicación adquirió mediante la conexión y todos los recursos que se utilizaron para crear y mantener la conexión. Por ejemplo, si el proceso de aplicación ejecuta una sentencia `RELEASE`, los cursores abiertos se cerrarán al finalizar la conexión durante la siguiente operación de confirmación.

Una conexión también puede finalizar a causa de una anomalía en las comunicaciones. Si la conexión que finaliza es la actual, el proceso de aplicación pasa al estado de no conectado.

Todas las conexiones de un proceso de aplicación finalizan al concluir el proceso.

Opciones que controlan la semántica de la unidad de trabajo distribuida

La semántica de gestión de la conexión de tipo 2 viene determinada por un conjunto de opciones del precompilador. A continuación encontrará un resumen de estas opciones, con los valores por omisión indicados con texto en negrita y subrayado. Para obtener detalles consulte los manuales *Consulta de mandatos* o *Administrative API Reference*.

- `CONNECT` (**1** | 2)
Especifica si las sentencias `CONNECT` se procesarán como tipo 1 o como tipo 2.
- `SQLRULES` (**DB2** | `STD`)
Especifica si las sentencias `CONNECT` de tipo 2 deben procesarse según las reglas de `DB2`, que permiten que `CONNECT` conmute a una conexión inactiva, o según las reglas `SQL92 Standard (STD)`, que no permiten esta posibilidad.
- `DISCONNECT` (**EXPLICIT** | `CONDITIONAL` | `AUTOMATIC`)
Especifica las conexiones de la base de datos que se desconectan cuando se produce una operación de confirmación. Son:
 - las que se han marcado explícitamente para su liberación mediante la sentencia de `SQL RELEASE (EXPLICIT)`, o bien,
 - las que no tienen ningún cursor `WITH HOLD` abierto así como las marcadas para liberación (`CONDITIONAL`)⁷ o
 - todas las conexiones (`AUTOMATIC`).

7. La opción `CONDITIONAL` no funcionará correctamente con los servidores de nivel inferior anteriores a la Versión 2. En estos casos, se producirá una desconexión, independientemente de la presencia de cursores `WITH HOLD`

- SYNCPOINT (**ONEPHASE** | TWOPHASE | NONE)

Especifica el modo en que se van a coordinar las operaciones de confirmación o retrotracción en las conexiones de varias bases de datos.

ONEPHASE Las actualizaciones sólo pueden producirse en una base de datos de la unidad de trabajo y el resto de las bases de datos son de sólo lectura. Cualquier intento de actualización en otra base de datos producirá un error (SQLSTATE 25000).

TWOPHASE Se utilizará un Gestor de transacciones (TM) en tiempo de ejecución para coordinar las confirmaciones en dos fases en todas las bases de datos que den soporte a este protocolo.

NONE No utiliza ningún TM para realizar la confirmación en dos fases y no impone ningún actualizador único, lector múltiple. Cuando se ejecuta una sentencia COMMIT o ROLLBACK, las sentencias COMMIT o ROLLBACK individuales se envían a todas las bases de datos. Si hay una o varias operaciones de retrotracción anómalas, se produce un error (SQLSTATE 58005). Si hay una o más operaciones de confirmación anómalas, se produce un error (SQLSTATE 40003).

Para alterar temporalmente cualquiera de las opciones anteriores en tiempo de ejecución, utilice una interfaz de programación de aplicaciones (API) especial SET CLIENT. Sus valores actuales se pueden obtener mediante la API especial QUERY CLIENT. Recuerde que no se trata de sentencias de SQL, sino que son unas API definidas en diversos lenguajes principales y en el Procesador de línea de mandatos. En los manuales *Consulta de mandatos* y *Administrative API Reference* puede encontrar su definición.

Consideraciones acerca de la representación de datos

Los diversos sistemas representan los datos de maneras distintas. Cuando se mueven datos de un sistema a otro, a veces debe realizarse una conversión de datos. Los productos que soportan DRDA realizan automáticamente las conversiones necesarias en el sistema receptor. Para realizar la conversión con datos numéricos, el sistema necesita conocer el tipo de datos y el modo en que el sistema emisor representa dicho tipo de datos. Con datos de caracteres, se necesita información adicional para convertir las series de caracteres. La conversión de series depende de la página de códigos de los datos y de la operación que se ha de realizar con dichos datos. Las conversiones de caracteres se llevan a cabo de acuerdo con la Arquitectura de representación de datos de caracteres (CDRA) de IBM. Para obtener más información sobre la conversión de caracteres, consulte el documento *Character Data Representation Architecture: Reference & Registry* SC09-2190-00.

Sistemas federados DB2

Esta sección proporciona una visión general de los elementos de un sistema federado DB2, una visión general de las tareas que los administradores y usuarios del sistema realizan y explicaciones de los conceptos asociados a estas tareas.

Servidor federado, base de datos federada y fuentes de datos

Un *sistema federado DB2* es un sistema informático distribuido que consta de un servidor DB2 y de múltiples fuentes de datos.

- Un servidor DB2 de un sistema federado recibe el nombre de *servidor federado*.

En una instalación DB2, puede configurarse cualquier número de instancias de DB2 para que funcionen como servidores federados.

- En un sistema federado, el servidor federado envía consultas a múltiples fuentes de datos.

Cada fuente de datos consta de una instancia RDBMS y de una o más bases de datos soportadas por la instancia. Las fuentes de datos de un sistema federado DB2 pueden incluir instancias de Oracle e instancias de los miembros de la familia DB2.

Las fuentes de datos son semiautónomas. Por ejemplo, el servidor federado puede enviar consultas a fuentes de datos Oracle al mismo tiempo que aplicaciones Oracle acceden a estas fuentes de datos. Un sistema federado DB2 no monopoliza ni restringe el acceso a fuentes de datos Oracle ni de otra clase (fuera de las restricciones de integridad y de bloqueo).

Para los usuarios finales y aplicaciones cliente, las fuentes de datos aparecen como una sola base de datos colectiva. En realidad, los usuarios y aplicaciones intercambian información con una base de datos, denominada *base de datos federada*, que está en el servidor federado. Para obtener datos de las fuentes de datos, someten consultas en SQL de DB2 a la base de datos federada. Después DB2 distribuye las consultas a las fuentes de datos adecuadas. DB2 también proporciona planes de acceso para optimizar las consultas (en algunos casos, estos planes llaman al servidor federado para procesar las consultas en lugar de llamar a la fuente de datos). Finalmente, DB2 reúne los datos solicitados y los pasa a los usuarios y aplicaciones.

Las consultas sometidas desde el servidor federado a las fuentes de datos deben ser de sólo lectura. Para grabar en una fuente de datos (por ejemplo, para actualizar una tabla de fuente de datos), los usuarios y las aplicaciones deben utilizar el SQL propio de la fuente de datos en una modalidad especial llamada *paso a través*.

Tareas realizadas en un sistema federado DB2

Esta sección presenta los conceptos asociados con las tareas de usuario necesarias para establecer y utilizar un sistema federado. (En esta sección y en

las siguientes, el término *usuarios* se refiere a todo tipo de personal que trabaja con sistemas federados, por ejemplo administradores de bases de datos, programadores de aplicaciones y usuarios finales).

La lista siguiente de tareas identifica los tipos de usuarios que ejecutan normalmente las tareas. También pueden realizar dichas tareas otros tipos de usuarios. Por ejemplo, la lista indica que, normalmente, los DBA crean correlaciones entre las autorizaciones para acceder a la base de datos federada y las autorizaciones para acceder a las fuentes de datos. Sin embargo, los programadores de aplicaciones y los usuarios finales también pueden ejecutar dicha tarea.

Para establecer y utilizar un sistema federado DB2:

1. El DBA designa un servidor DB2 como servidor federado. Consulte la publicación *Suplemento de instalación y configuración* para obtener información sobre cómo se realiza esta tarea.
2. El DBA configura las fuentes de datos para el acceso del modo siguiente:
 - a. El DBA se conecta a la base de datos federada.
 - b. El DBA crea un reiniciador para cada categoría de fuente de datos que se ha de incluir en el sistema federado. Los *reiniciadores* son mecanismos mediante los cuales el servidor federado interactúa con las fuentes de datos. Consulte el apartado “Reiniciadores y módulos de reiniciador” en la página 59.
 - c. El DBA suministra al servidor federado una descripción de cada fuente de datos. La descripción se denomina *definición de servidor*. Consulte el apartado “Definiciones de servidor y opciones de servidor” en la página 59.
 - d. Si el ID de autorización de un usuario utilizado para acceder a la base de datos federada es diferente del ID de autorización utilizado para acceder a una fuente de datos, el DBA define una asociación entre los dos ID de autorización. Esta asociación se denomina *correlación de usuarios*. Consulte el apartado “Correlaciones de usuarios y opciones de usuario” en la página 62.
 - e. Si una correlación por omisión entre un tipo de datos DB2 y el tipo de datos de una fuente de datos no satisface las necesidades del usuario, el DBA modifica la correlación según sea necesario. Una *correlación de tipo de datos* es una asociación definida entre dos tipos de datos compatibles — uno soportado por la base de datos federada y otro soportado por una fuente de datos. Consulte el apartado “Correlaciones de tipos de datos” en la página 62.
 - f. Si una correlación por omisión entre una función DB2 y una función de una fuente de datos no satisface las necesidades del usuario, el DBA modifica la correlación según sea necesario. Una *correlación de funciones* es una asociación definida entre dos funciones compatibles — una

soportada por la base de datos federada y otra soportada por una fuente de datos. Consulte el apartado “Correlaciones de funciones, plantillas de funciones y opciones de correlación de funciones” en la página 63.

- g. Los DBA y los programadores de aplicaciones crean apodos para las tablas y las vistas de fuentes de datos que se han de acceder. Un *apodo* es un identificador mediante el cual el sistema federado hace referencia a una vista o tabla de fuente de datos. Consulte el apartado “Apodos y opciones de columna” en la página 64.
 - h. Opcional: Si una tabla de fuente de datos no tiene índice, el DBA puede proporcionar al servidor federado la misma clase de información que la que podría contener la definición de un índice real. Si una tabla de fuente de datos tiene un índice que el servidor federado no conoce, el DBA puede informar al servidor de la existencia del índice. En cualquier caso, la información que el DBA suministra ayuda a DB2 a optimizar las consultas de los datos de tabla. Esta información se denomina *especificación de índice*. Consulte el apartado “Especificaciones de índice” en la página 65.
3. Los programadores de aplicaciones y los usuarios finales recuperan información de las fuentes de datos:
- Mediante la utilización del SQL para DB2, los programadores de aplicaciones y los usuarios finales consultan tablas y vistas que son referidas por apodos. Las consultas dirigidas a dos o más fuentes de datos se denominan *consultas distribuidas*. Consulte el apartado “Peticiónes distribuidas” en la página 66 para obtener detalles.
En el proceso de una consulta, el servidor federado puede realizar operaciones que están soportadas por el SQL de DB2 pero no por el SQL de la fuente de datos. Esta posibilidad se denomina *compensación*. Consulte el apartado “Compensación” en la página 67.
 - Los programadores de aplicaciones y los usuarios finales someten a veces consultas, sentencias DML⁸ y sentencias DDL⁹ en fuentes de datos con el formato específico de SQL utilizado por la fuente de datos, en lugar del formato de SQL utilizado por DB2. Los programadores y usuarios puede realizar dicha acción en modalidad de paso a través. Consulte el apartado “Paso a través” en la página 68.

Las secciones siguientes explican los conceptos mencionados en esta lista de tareas en el mismo orden en el que aparecen en la lista. Algunas de estas secciones también introducen conceptos relacionados.

8. DML (Data Manipulation Language - Lenguaje de manipulación de datos) es el subconjunto de sentencias de SQL que se utilizan para manipular datos.

9. DDL (Data Definition Language - Lenguaje de definición de datos) es el subconjunto de sentencias de SQL que se utilizan para describir las relaciones de los datos de una base de datos.

Reiniciadores y módulos de reiniciador

Un reiniciador es el mecanismo mediante el que el servidor federado se comunica con una fuente de datos y recupera datos de ella. Para implantar un reiniciador, el servidor utiliza las rutinas almacenadas en una biblioteca denominada *módulo de reiniciador*. Estas rutinas permiten al servidor realizar operaciones como, por ejemplo, conectarse a una fuente de datos y recuperar datos de la misma repetidamente.

Existen tres tipos de reiniciadores:

- Se utiliza un reiniciador con el nombre por omisión DRDA para todas las fuentes de datos de la familia DB2.
- Se utiliza un reiniciador con el nombre por omisión SQLNET para todas las fuentes de datos Oracle soportadas por el software de cliente SQL*Net de Oracle.
- Se utiliza un reiniciador con el nombre por omisión NET8 para todas las fuentes de datos de Oracle soportadas por el software de cliente Net8 de Oracle.

Un reiniciador se registra en el servidor federado con la sentencia CREATE WRAPPER. Vea “CREATE WRAPPER” en la página 974.

Definiciones de servidor y opciones de servidor

Después de que el DBA registra un reiniciador que permite al servidor federado interactuar con las fuentes de datos, el DBA define estas fuentes de datos en la base de datos federada. Esta sección:

- Distingue los diferentes significados del término “servidor”.
- Describe la definición que proporciona el DBA.
- Describe el SQL para especificar determinados parámetros, denominados *opciones de servidor*, que contienen partes de una definición de servidor.

Tres significados de “servidor”

En los términos *definición de servidor* y *opción de servidor* así como en las sentencias de SQL descritas en las secciones siguientes, la palabra *servidor* sólo hace referencia a las fuentes de datos. No hace referencia al servidor federado ni a los servidores de aplicaciones DB2.

Sin embargo, los conceptos de servidores de aplicaciones DB2 y servidores federados se superponen. Tal como se indica en el apartado “Base de datos relacional distribuida” en la página 43, un servidor de aplicaciones es una instancia del gestor de bases de datos a la que se conectan los procesos de aplicaciones y someten peticiones. Esto también se cumple para un servidor federado; por lo tanto, un servidor federado es un tipo de servidor de aplicaciones. Pero dos puntos importantes lo distinguen de otros servidores de aplicaciones:

- Está configurado para recibir peticiones que están pensadas en última instancia para fuentes de datos y distribuye estas peticiones a las fuentes de datos.
- Al igual que otros servidores de aplicaciones, un servidor federado utiliza protocolos de comunicaciones DRDA para comunicarse con las instancias de la familia DB2. A diferencia de otros servidores de aplicaciones, un servidor federado utiliza los protocolos de comunicaciones SQLNET y net8 para comunicarse con las instancias de Oracle.

Introducción a las definiciones de servidor

Cuando se define una fuente de datos en la base de datos federada, el DBA proporciona un nombre para la fuente de datos así como la información que pertenece a la fuente de datos. Esta información incluye el tipo y la versión del RDBMS del cual la fuente de datos es una instancia y el nombre de RDBMS para la fuente de datos. También incluye los metadatos que son específicos de RDBMS. Por ejemplo, una fuente de datos de la familia DB2 puede tener múltiples bases de datos y la definición de esa fuente de datos debe especificar a qué base de datos puede conectarse un servidor federado. Por el contrario, una fuente de datos Oracle tiene una base de datos y el servidor federado puede conectarse a la base de datos sin necesidad de conocer su nombre. Por consiguiente, el nombre no se incluye en la definición de servidor federado de la fuente de datos.

El nombre y la información que el DBA suministra se denomina colectivamente definición de servidor. Este término refleja el hecho de que las fuentes de datos responden a peticiones de datos y que, por lo tanto, son servidores por derecho propio. Otros términos también reflejan este hecho. Por ejemplo:

- Parte de la información de una definición de servidor se almacena como opciones de servidor. De este modo, el nombre de una fuente de datos se almacena como un valor de una opción de servidor denominada NODE. Para una fuente de datos de la familia DB2, el nombre de la base de datos a la que se conecta el servidor federado se almacena como un valor de una opción de servidor denominada DBNAME.
- Las sentencias de SQL para la creación y modificación de una definición de servidor se denominan CREATE SERVER y ALTER SERVER, respectivamente.

Sentencias de SQL para establecer opciones de servidor

Los valores se asignan a las opciones de servidor a través de las sentencias CREATE SERVER, ALTER SERVER y SET SERVER OPTION.

Las sentencias CREATE SERVER y ALTER SERVER establecen las opciones de servidor en valores que persisten a través de sucesivas conexiones a la fuente de datos. Estos valores se almacenan en el catálogo. Considere este caso: Un DBA del sistema federado utiliza la sentencia CREATE SERVER para definir

una nueva fuente de datos Oracle en el sistema federado. La base de datos de esta fuente de datos utiliza el mismo orden de clasificación que el utilizado por la base de datos federada. El DBA desea que el optimizador conozca esta coincidencia, para que la aproveche para mejorar el rendimiento. De acuerdo con ello, en la sentencia CREATE SERVER, el DBA establece una opción de servidor llamada COLLATING_SEQUENCE en 'Y' (para indicar que el orden de clasificación de la fuente de datos y el de la base de datos federada son iguales). El valor 'Y' se anota en el catálogo y permanece en vigor mientras los usuarios y las aplicaciones acceden a la fuente de datos Oracle.

Unos meses más tarde, el DBA de Oracle cambia el orden de clasificación de la fuente de datos de Oracle utilizando la sentencia ALTER SERVER. Por consiguiente, el DBA del sistema federado restablece COLLATING_SEQUENCE en 'N' (para indicar que el orden de clasificación de la fuente de datos no es igual al de la base de datos federada). Se actualiza el catálogo y el nuevo valor permanece en vigor mientras los usuarios y las aplicaciones continúan accediendo a la fuente de datos.

La sentencia SET SERVER OPTION altera temporalmente el valor de una opción de servidor durante una sola conexión a la base de datos federada. El valor que prevalece no se almacena en el catálogo.

Como ilustración: Una opción de servidor denominada PLAN_HINTS puede establecerse en un valor que permita que DB2 suministre a las fuentes de datos Oracle fragmentos de sentencias, denominados indicaciones de planes, que ayudan a los optimizadores de Oracle a realizar su trabajo. Por ejemplo, las indicaciones de planes pueden ayudar a un optimizador a decidir qué índice debe utilizar en el acceso a una tabla o qué secuencia de unión de tablas debe utilizar para recuperar los datos de un conjunto resultante.

Para las fuentes de datos ORACLE1 y ORACLE2, se establece la opción de servidor PLAN_HINTS en el valor por omisión, 'N' (no proporcione indicaciones de planes a estas fuentes de datos). A continuación, un programador escribe una petición distribuida de datos de ORACLE1 y ORACLE2, esperando que las indicaciones de planes ayuden a los optimizadores de estas fuentes de datos a mejorar las estrategias para acceder a dichos datos. De acuerdo con ello, el programador utiliza la sentencia SET SERVER OPTION para alterar temporalmente el 'N' y convertirlo en 'Y' (indicar que se deben proporcionar indicaciones de planes). El 'Y' sólo permanece en vigor mientras la aplicación con la petición está conectada a la base de datos federada; no se almacena en el catálogo.

Consulte los apartados "CREATE SERVER" en la página 831, "ALTER SERVER" en la página 561 y "SET SERVER OPTION" en la página 1186 para

obtener más información. Consulte el apartado “Opciones de servidor” en la página 1415 para obtener descripciones de todas las opciones de servidor y sus valores.

Correlaciones de usuarios y opciones de usuario

El servidor federado puede enviar la petición distribuida de una aplicación o un usuario autorizado a una fuente de datos bajo una de estas condiciones o bajo ambas:

- El usuario o la aplicación utilizan el mismo ID de usuario tanto para la base de datos federada como para la fuente de datos. Además, si la fuente de datos necesita una contraseña, el usuario o la aplicación utilizan la misma contraseña para la base de datos federada que para la fuente de datos.
- La autorización del usuario o de la aplicación para acceder a la base de datos federada difiere en alguna manera de la autorización del usuario o de la aplicación para acceder a la fuente de datos. Además, cuando el usuario o la aplicación piden acceso a la fuente de datos, se cambia la autorización de la base de datos federada por la autorización de la fuente de datos, para que pueda otorgarse el acceso. Este cambio sólo puede producirse si existe una asociación definida, denominada correlación de usuarios, entre las dos autorizaciones.

Las correlaciones de usuarios pueden definirse y modificarse con las sentencias `CREATE USER MAPPING` y `ALTER USER MAPPING`. Estas sentencias incluyen parámetros, denominados *opciones de usuario*, a los que se asignan valores relacionados con la autorización. Por ejemplo, suponga que un usuario tiene el mismo ID, pero distintas contraseñas, para la base de datos federada y una fuente de datos. Para que el usuario acceda a la fuente de datos, es necesario correlacionar las contraseñas entre sí. Esto se puede realizar con una sentencia `CREATE USER MAPPING` en la cual la contraseña de la fuente de datos se asigna como valor a una opción de usuario denominada `REMOTE_PASSWORD`.

Consulte el apartado “`CREATE USER MAPPING`” en la página 955, el apartado “`ALTER USER MAPPING`” en la página 610 y la publicación *Administration Guide* para obtener más información. Consulte el apartado “Opciones de usuario” en la página 1420 para obtener descripciones de las opciones de usuario y sus valores.

Correlaciones de tipos de datos

Para que el servidor federado recupere los datos de las columnas de las vistas y las tablas de las fuentes de datos, los tipos de datos de las columnas de la fuente de datos deben correlacionarse con los tipos de datos correspondientes ya definidos en la base de datos federada. DB2 suministra las correlaciones por omisión para la mayoría de las clases de tipos de datos. Por ejemplo, el tipo `FLOAT` de Oracle se correlaciona por omisión con el tipo `DOUBLE` de DB2 y el tipo `DATE` de DB2 Universal Database para OS/390 se correlaciona

por omisión con el tipo DATE de DB2. No hay ninguna correlación para los tipos de datos que los servidores federados de DB2 no soportan: LONG VARCHAR, LONG VARGRAPHIC, DATALINK, tipos de objeto grande (LOB) y tipos definidos por el usuario.

Consulte el apartado “Correlaciones de tipos de datos por omisión” en la página 1421 para obtener los listados de las correlaciones de tipos de datos por omisión.

Cuando se devuelven los valores de una columna de la fuente de datos, se ajustan completamente al tipo DB2 de la correlación de tipos que se aplica a la columna. Si esta correlación es un valor por omisión, los valores también se ajustan completamente al tipo de la fuente de datos de la correlación. Por ejemplo, cuando se define una tabla Oracle con una columna FLOAT en la base de datos federada, se aplicará automáticamente la correlación por omisión FLOAT de Oracle con DOUBLE de DB2 a esta columna, a menos que se haya alterado temporalmente. En consecuencia, los valores devueltos de la columna se ajustarán completamente a FLOAT y DOUBLE.

Es posible cambiar el formato o la longitud de los valores devueltos cambiando el tipo de DB2 al que los valores deben ajustarse. Por ejemplo, el tipo DATE de Oracle sirve para las indicaciones de la hora. Por omisión, se correlaciona con el tipo TIMESTAMP de DB2. Suponga que varias columnas de la tabla Oracle tienen un tipo DATE de datos y que un usuario desea que las consultas de estas columnas indiquen sólo la hora. El usuario puede correlacionar el tipo DATE de Oracle con el tipo TIME de DB2, alterando temporalmente el valor por omisión. De este modo, cuando se consulten las columnas, sólo se devolverá la parte de la hora de las indicaciones de fecha y hora.

Se puede utilizar la sentencia CREATE TYPE MAPPING para crear una correlación de tipos de datos modificada que se aplique a una o varias fuentes de datos. Se puede utilizar la sentencia ALTER NICKNAME para modificar una correlación de tipos de datos para una columna específica de una tabla específica.

Consulte el apartado “CREATE TYPE MAPPING” en la página 950, el apartado “ALTER NICKNAME” en la página 548 y la publicación *Application Development Guide* para obtener más información.

Correlaciones de funciones, plantillas de funciones y opciones de correlación de funciones

Para que el servidor federado reconozca una función de fuente de datos, la función debe correlacionarse con una función DB2 existente en el servidor. DB2 suministra correlaciones por omisión entre las funciones de fuente de datos incorporadas existentes y las funciones DB2 incorporadas. Si un usuario

desea utilizar una función de fuente de datos que el servidor federado no reconoce (por ejemplo, una nueva función incorporada o una función definida por el usuario), el usuario debe crear una correlación entre esta función y una función complementaria situada en la base de datos federada. Si no existe ninguna función complementaria, el usuario debe crear una que cumpla los requisitos siguientes:

- Si la función de fuente de datos tiene parámetros de entrada, la función complementaria debe tener el mismo número de parámetros de entrada que la función de fuente de datos. Si la función de fuente de datos no tiene ningún parámetro de entrada, la función complementaria no puede tener ninguno.
- Los tipos de datos para los parámetros de entrada (si existen) y los valores devueltos para la función complementaria deben ser compatibles con los tipos de datos correspondientes de la función de fuente de datos.

La función complementaria puede ser una función completa o una plantilla de función. Una *plantilla de función* es una función parcial que no tiene código ejecutable. No se puede invocar independientemente; su única finalidad es participar en una correlación con una función de fuente de datos, para que la función de fuente de datos pueda invocarse desde el servidor federado.

Las correlaciones de funciones se crean con la sentencia CREATE FUNCTION MAPPING. Esta sentencia incluye parámetros, llamados *opciones de correlación de funciones*, a los que el usuario puede asignar valores que pertenecen a la correlación que se está creando o a la función de fuente de datos de la correlación. Dichos valores pueden incluir, por ejemplo, las estadísticas estimadas sobre la actividad general que se consumirá cuando se invoque la función de fuente de datos. El optimizador utiliza estas estimaciones en el desarrollo de estrategias para invocar la función.

Consulte el apartado “CREATE FUNCTION (fuente o plantilla)” en la página 750 y el apartado “CREATE FUNCTION MAPPING” en la página 769 para obtener detalles acerca de la creación de plantillas de funciones y correlaciones de funciones. Consulte el apartado “Opciones de correlación de funciones” en la página 1414 para obtener descripciones de las opciones de correlación de funciones y sus valores. Consulte la publicación *Application Development Guide* para conocer las directrices para optimizar la invocación de las funciones de fuente de datos.

Apodos y opciones de columna

Cuando una aplicación cliente somete una petición distribuida al servidor federado, el servidor divide la petición entre las fuentes de datos adecuadas. La petición no necesita especificar estas fuentes de datos. En lugar de ello, hace referencia a las vistas y las tablas de la fuente de datos mediante apodos, los cuales se correlacionan con los nombres de vistas y tablas de la fuente de datos. Las correlaciones eliminan la necesidad de calificar los apodos

mediante nombres de fuentes de datos. Las ubicaciones de las tablas y las vistas son transparentes para la aplicación cliente.

Los apodos no son nombres alternativos para las tablas y las vistas del modo en que lo son los seudónimos; son punteros mediante los cuales el servidor federado hace referencia a estos objetos. Los apodos se definen con la sentencia `CREATE NICKNAME`. Consulte el apartado “`CREATE NICKNAME`” en la página 795 para obtener detalles.

Cuando se crea un apodo para una tabla o vista, el catálogo se rellena con metadatos que el optimizador puede utilizar para facilitar el acceso a la tabla o la vista. Por ejemplo, el catálogo se proporciona con los nombres de los tipos de datos DB2 con los que se correlacionan los tipos de datos de las columnas de la tabla o la vista. Si el apodo es para una tabla con un índice, el catálogo también se suministra con la información relativa al índice; por ejemplo, el nombre de cada columna de la clave de índice.

Después de crear un apodo, el usuario puede suministrar al catálogo más metadatos para el optimizador; por ejemplo, los metadatos que describen los valores de ciertas columnas de la tabla o la vista a la que el apodo hace referencia. El usuario asigna estos metadatos a parámetros denominados *opciones de columna*. Como ilustración: si una columna de tabla sólo contiene series numéricas, el usuario puede indicarlo asignando el valor ‘Y’ a una opción de columna denominada `NUMERIC_STRING`. Como resultado, el optimizador puede formar estrategias para clasificar estas series en la fuente de datos y ahorrar, de esta manera, la actividad general de llevarlas al servidor federado y clasificarlas allí. El ahorro es especialmente grande cuando la base de datos que contiene los valores tiene un orden de clasificación que difiere del orden de clasificación de la base de datos federada.

Las opciones de columna se definen con la sentencia `ALTER NICKNAME`. Consulte el apartado “`ALTER NICKNAME`” en la página 548 para obtener más información sobre esta sentencia. Consulte el apartado “Opciones de columna” en la página 1413 para obtener descripciones de las opciones de columna y sus valores.

Especificaciones de índice

Cuando se crea un apodo para una tabla de fuente de datos, el servidor federado suministra al catálogo información acerca de todos los índices que tenga la tabla de fuente de datos. El optimizador utiliza esta información para facilitar la recuperación de los datos de la tabla. No obstante, si la tabla no tiene índices, el usuario puede suministrar la información que normalmente contiene una definición de índice; por ejemplo, la columna o columnas de la

tabla en las que se ha de buscar para encontrar rápidamente la información. El usuario ejecuta una sentencia CREATE INDEX que contiene la información y hace referencia al apodo de la tabla.

El usuario puede suministrar al optimizador información similar para las tablas que tienen índices que el servidor federado no conoce. Por ejemplo, suponga que se crea el apodo NICK1 para una tabla que no tiene índice pero que adquiere uno posteriormente o que se crea el apodo NICK2 para una vista de una tabla que tiene un índice. En estas situaciones, el servidor federado no conocería los índices. Pero el usuario podría utilizar dos sentencias CREATE INDEX para informar al servidor de que los índices existen. Una sentencia hace referencia a NICK1 y contiene información acerca del índice de la tabla NICK1 identifica. La otra hace referencia a NICK2 y contiene información acerca del índice de la tabla de base que sirve de fundamento para la vista que NICK2 identifica.

En casos como los recién descritos, la información de la sentencia CREATE INDEX se cataloga como un conjunto de metadatos denominado especificación de índice. Tenga en cuenta que cuando la sentencia hace referencia a un apodo, sólo produce una especificación de índice, no un índice real. Consulte el apartado "CREATE INDEX" en la página 774 y la publicación *Administration Guide: Performance* para obtener más información.

Peticiones distribuidas

Una petición distribuida puede utilizar dispositivos, por ejemplo subconsultas y subselecciones de unión, para especificar las columnas de tabla o vista a las que se debe acceder y qué datos deben recuperarse.

Los ejemplos de esta sección se obtienen del escenario siguiente: un servidor federado está configurado para acceder a una fuente de datos DB2 Universal Database para OS/390, una fuente de datos DB2 Universal Database para AS/400 y una fuente de datos Oracle. Hay una tabla almacenada en cada fuente de datos que contiene la información de los empleados. El servidor federado hace referencia a estas tablas por apodos que hacen referencia al lugar en que residen las tablas: UDB390_EMPLOYEES, AS400_EMPLOYEES y ORA_EMPLOYEES. Además de la información de la tabla de empleados, la fuente de datos Oracle tiene una tabla que contiene información acerca de los países donde viven los empleados. El apodo para esta segunda tabla es ORA_COUNTRIES.

Petición con subconsulta

La tabla AS400_EMPLOYEES contiene los números de teléfono de los empleados que viven en Asia. También contiene los códigos de país asociados con estos números de teléfono, pero no lista los países que los códigos representan. Sin embargo, la tabla ORA_COUNTRIES, lista los códigos y los países. La consulta siguiente utiliza una subconsulta para averiguar el código

de país para China; y utiliza las cláusulas SELECT y WHERE para listar los empleados de AS400_EMPLOYEES cuyos números de teléfono necesiten este código en particular.

```
SELECT NAME, TELEPHONE
FROM DJADMIN.AS400_EMPLOYEES
WHERE COUNTRY_CODE IN
(SELECT COUNTRY_CODE
FROM DJADMIN.ORA_COUNTRIES
WHERE COUNTRY_NAME = 'CHINA')
```

Cuando se compila una petición distribuida como la anterior, el recurso de regrabación de consulta del compilador la transforma a un formato que pueda optimizarse más fácilmente.

Petición de unión

Una unión relacional produce un conjunto resultante que contiene una combinación de columnas recuperadas de dos o más tablas. Siempre se deberán especificar condiciones para limitar el tamaño de las filas del conjunto de resultados.

La consulta siguiente combina los nombres de los empleados y los nombres de sus países correspondientes comparando los códigos de país listados en dos tablas. Cada tabla reside en una fuente de datos diferente.

```
SELECT T1.NAME, T2.COUNTRY_NAME
FROM DJADMIN.UDB390_EMPLOYEES T1, DJADMIN.ORA_COUNTRIES T2
WHERE T1.COUNTRY_CODE = T2.COUNTRY_CODE
```

Compensación

La *compensación* es el proceso de sentencias de SQL para bases de datos relacionales que no soportan esas sentencias. Cada tipo de RDBMS DB2 Universal Database para AS/400, DB2 Universal Database para OS/390, Oracle, etc) soporta un subconjunto del estándar internacional de SQL. Además, algunos tipos soportan construcciones SQL que exceden este estándar. La totalidad de SQL que un tipo de RDBMS soporta se denomina *dialecto SQL*. Si una construcción SQL se encuentra en un dialecto SQL de DB2, pero no en un dialecto de la fuente de datos, el servidor federado puede implementar esta construcción en nombre de la fuente de datos.

Ejemplo 1: El SQL de DB2 incluye la cláusula expresión-tabla-común. En esta cláusula, se puede especificar un nombre mediante el cual todas las cláusulas FROM de una selección completa pueden hacer referencia a un conjunto resultante. El servidor federado procesará una expresión-tabla-común para una base de datos Oracle aunque el dialecto SQL de Oracle no incluya la expresión-tabla-común.

Ejemplo 2: Al conectarse a una fuente de datos que no soporta múltiples cursores abiertos en una aplicación, el servidor federado puede simular esta función estableciendo conexiones independientes, simultáneas con la fuente de datos. De manera similar, el servidor federado puede simular la posibilidad CURSOR WITH HOLD para una fuente de datos que no proporcione esa función.

La compensación hace que sea posible utilizar el dialecto SQL de DB2 para realizar todas las consultas soportadas por el servidor federado. No es necesario utilizar los dialectos específicos para los RDBMS distintos de DB2.

Paso a través

Los usuarios pueden utilizar la función de paso a través para comunicarse con las fuentes de datos en el dialecto SQL de las fuentes de datos. En la modalidad de paso a través, los usuarios no sólo pueden someter consultas, sino también sentencias DML y DDL. Consulte el apartado “Proceso SQL en sesiones de paso a través” en la página 1423 para obtener información sobre cómo DB2 y las fuentes de datos gestionan el proceso de sentencias sometidas en sesiones de paso a través.

El servidor federado proporciona las siguientes sentencias de SQL para gestionar sesiones de paso a través:

SET PASSTHRU

Inicia y termina sesiones de paso a través.

GRANT (Privilegios de servidor)

Otorga a un usuario, grupo, lista de ID de autorización o PUBLIC la autorización para iniciar sesiones de paso a través para una fuente de datos específica.

REVOKE (Privilegios del servidor)

Revoca la autorización para iniciar sesiones de paso a través.

Se aplican ciertas restricciones en la utilización del paso a través. Por ejemplo, en una sesión de paso a través, no se puede abrir un cursor directamente para un objeto de fuente de datos. Consulte el apartado “Consideraciones y restricciones” en la página 1424 para obtener una lista completa de las restricciones.

Capítulo 3. Elementos del lenguaje

Este capítulo define la sintaxis básica del SQL y los elementos del lenguaje comunes a muchas sentencias de SQL.

Materia	Página
Caracteres	69
Símbolos	70
Identificadores	71
Convenios de denominación y calificaciones implícitas de nombres de objetos	72
Seudónimos	78
ID de autorización y nombres-autorización	79
Tipos de datos	82
Promoción de los tipos de datos	100
Conversión entre tipos de datos	101
Asignaciones y comparaciones	105
Reglas para los tipos de datos del resultado	120
Constantes	129
Registros especiales	132
Nombres de columna	143
Referencias a variables del lenguaje principal	151
Funciones	159
Métodos	168
Semántica de enlace conservador	174
Expresiones	176
Predicados	212
Condiciones de búsqueda	232

Caracteres

Los símbolos básicos de las palabras clave y de los operadores del lenguaje SQL son caracteres de un solo byte que forman parte de todos los juegos de caracteres IBM. Los caracteres del lenguaje se clasifican en letras, dígitos y caracteres especiales.

Caracteres

Una *letra* es cualquiera de las 26 letras mayúsculas (A a Z) y 26 letras minúsculas (a a z) más los tres caracteres (\$, # y @), que se incluyen para la compatibilidad con los productos de base de datos de sistema principal (por ejemplo, en la página de códigos 850, \$ está en la posición X'24', # está en la posición X'23' y @ está en la posición X'40'). Las letras también incluyen los caracteres alfabéticos de los juegos de caracteres ampliados. Los juegos de caracteres ampliados contienen caracteres alfabéticos adicionales, tales como caracteres con signos diacríticos (´ es un ejemplo de signo diacrítico). Los caracteres disponibles dependen de la página de códigos que se utiliza.

Un *dígito* es cualquier carácter del 0 al 9.

Un *carácter especial* es cualquiera de los caracteres listados a continuación:

	blanco	–	signo menos
"	comillas o comillas dobles	.	punto
%	tanto por ciento	/	barra inclinada
&	símbolo para "y"	:	dos puntos
'	apóstrofo o comillas simples	;	punto y coma
(abrir paréntesis	<	menor que
)	cerrar paréntesis	=	igual
*	asterisco	>	mayor que
+	signo más	?	signo de interrogación
,	coma	~	subrayado
	barra vertical		signo de intercalación
!	signo de admiración		

Consideraciones acerca de MBCS

Todos los caracteres de múltiples bytes se tratan como letras, excepto el blanco de doble byte que es un carácter especial.

Símbolos

Las unidades sintácticas básicas del lenguaje son los *símbolos*. Un símbolo es una secuencia de uno o varios caracteres. Un símbolo no puede contener caracteres en blanco, a menos que sea una constante de tipo serie o un identificador delimitado, que pueden contener blancos. (Estos términos se definen más adelante.)

Los símbolos se clasifican en *ordinarios* y *delimitadores*:

- Un *símbolo ordinario* es una constante numérica, un identificador ordinario, un identificador del lenguaje principal o una palabra clave.

Ejemplos

1 .1 +2 SELECT E 3

- Un *símbolo delimitador* es una constante de tipo serie, un identificador delimitado, un símbolo de operador o cualquier carácter especial mostrado en los diagramas de sintaxis. Un signo de interrogación también es un símbolo delimitador cuando actúa como marcador de parámetros, tal como se explica en "PREPARE" en la página 1099.

Ejemplos

, 'serie' "fld1" = .

Espacios: Un espacio es una secuencia de uno o varios caracteres en blanco. Los símbolos que no son constantes de tipo serie ni identificadores delimitados no deben incluir ningún espacio. Los símbolos pueden ir seguidos de un espacio. Cada símbolo ordinario debe ir seguido por un espacio o por un símbolo delimitador si lo permite la sintaxis.

Comentarios: Las sentencias de SQL estático pueden incluir comentarios del lenguaje principal o comentarios de SQL. Se puede especificar cualquiera de estos tipos de comentario dondequiera que se pueda especificar un espacio, excepto dentro de un símbolo delimitador o entre las palabras clave EXEC y SQL. Los comentarios de SQL comienzan con dos guiones consecutivos (--) y finalizan con el final de la línea. Para obtener más información, vea "comentarios de SQL" en la página 544.

Mayúsculas y minúsculas: Los símbolos pueden incluir letras minúsculas, pero las letras minúsculas de un símbolo ordinario se convierten a mayúsculas, excepto en las variables del lenguaje principal en C, que tienen identificadores sensibles a las mayúsculas y minúsculas. Los símbolos delimitadores no se convierten nunca a mayúsculas. Por lo tanto, la sentencia:

```
select * from EMPLOYEE where lastname = 'Smith';
```

después de la conversión, es equivalente a:

```
SELECT * FROM EMPLOYEE WHERE LASTNAME = 'Smith';
```

Consideraciones acerca de MBCS

Las letras alfabéticas de múltiples bytes no se convierten a mayúsculas. Los caracteres de un solo byte, de la a a la z, se convierten a mayúsculas.

Identificadores

Un *identificador* es un símbolo que se utiliza para formar un nombre. En una sentencia de SQL, un identificador es un identificador SQL o un identificador del lenguaje principal.

Identificadores

Identificadores SQL

Existen dos tipos de identificadores SQL: *ordinarios* y *delimitados*

- Un *identificador ordinario* es una letra seguida de cero o varios caracteres, cada uno de los cuales es una letra mayúscula, un dígito o un carácter de subrayado. Un identificador ordinario no debe ser idéntico a una palabra reservada (consulte el “Apéndice H. Nombres de esquema reservados y palabras reservadas” en la página 1447 para obtener información sobre palabras reservadas).
- Un *identificador delimitado* es una secuencia de uno o varios caracteres entre comillas (“”). Dos comillas consecutivas se utilizan para representar unas comillas dentro del identificador delimitado. De esta manera un identificador puede incluir letras en minúsculas.

Algunos ejemplos de identificadores ordinarios y delimitados son:

WKLYSAL WKLY_SAL "WKLY_SAL" "WKLY SAL" "UNION" "wkly_sal"

Las conversiones de caracteres entre los identificadores creados en una página de códigos de doble byte pero utilizados por una aplicación o una base de datos de una página de códigos de múltiples bytes pueden necesitar una consideración especial. Después de una conversión a múltiples bytes, es posible que dichos identificadores excedan el límite de longitud de un identificador (consulte el “Apéndice O. Consideraciones acerca de EUC de japonés y de chino tradicional” en la página 1511 para obtener detalles).

Identificadores de sistema principal

Un *identificador de sistema principal* es un nombre declarado en el programa del sistema principal. Las reglas para formar un identificador del sistema principal son las reglas del lenguaje principal. Un identificador del sistema principal no debe tener más de 255 caracteres y no debe comenzar con los caracteres ‘SQL’ o ‘DB2’ en mayúsculas ni minúsculas.

Convenios de denominación y calificaciones implícitas de nombres de objetos

Las reglas para formar un nombre dependen del tipo de objeto designado por el nombre. Los nombres de los objetos de una base de datos pueden constar de un solo identificador o pueden ser objetos calificados con un esquema que consten de dos identificadores. Los nombres de los objetos calificados con un esquema pueden estar especificados sin nombre de esquema. En tales casos, está implícito un nombre de esquema.

En las sentencias de SQL dinámico, un nombre de objeto calificado con un esquema utiliza implícitamente el valor de registro especial CURRENT SCHEMA como calificador para las referencias al nombre de objeto no calificadas. Por omisión, se establece en el ID de autorización actual. Consulte el apartado “SET SCHEMA” en la página 1184 para obtener detalles. Si la

sentencia de SQL dinámico procede de un paquete enlazado con la opción DYNAMICRULES BIND, el registro especial CURRENT SCHEMA no se utiliza en la calificación. En un paquete DYNAMICRULES BIND, el calificador por omisión del paquete se utiliza como el valor para la calificación de referencias al objeto no calificadas dentro de sentencias de SQL dinámico.

En las sentencias de SQL estático, la opción de precompilación/enlace lógico QUALIFIER especifica implícitamente el calificador para los nombres de objetos de base de datos no calificados. Por omisión, se establece en el ID de autorización del paquete. Consulte el manual *Consulta de mandatos* para obtener detalles.

Los nombres de objeto siguientes, cuando se utilizan en el contexto de un procedimiento de SQL, sólo pueden utilizar los caracteres permitidos en un identificador ordinario, incluso aunque los nombres estén delimitados:

- nombre-condición
- etiqueta
- nombre-parámetro
- nombre-procedimiento
- nombre-variable-SQL
- nombre-sentencia

Los diagramas de sintaxis utilizan distintos términos para tipos diferentes de nombres. La lista siguiente define dichos términos. Para conocer la longitud máxima de varios identificadores, consulte el "Apéndice A. Límites de SQL" en la página 1253.

nombre-seudónimo	Nombre calificado con esquema que designa un seudónimo.
nombre-atributo	Identificador que designa un atributo de un tipo de datos estructurados.
nombre-autorización	Identificador que designa un usuario o un grupo. Tenga en cuenta que se pueden utilizar las siguientes restricciones en los caracteres: <ul style="list-style-type: none">• Los caracteres válidos van de la A a la Z, de la "a" a la "z", de 0 a 9, #, @, \$ y _.• El nombre no debe empezar por los caracteres 'SYS', 'IBM' o 'SQL'.• El nombre no debe ser: ADMINS, GUESTS, LOCAL, PUBLIC ni USERS.• Un ID de autorización delimitado no debe contener letras en minúsculas.

Convenios de denominación

nombre-agrupalmacinter	Identificador que designa una agrupación de almacenamientos intermedios.
nombre-columna	Nombre calificado o no calificado que designa una columna de una tabla o de una vista. El calificador es un nombre-tabla, nombre-vista, un apodo o un nombre-correlación.
nombre-condición	Identificador que designa una condición en un procedimiento SQL.
nombre-restricción	Identificador que designa una restricción de referencia, una restricción de clave primaria, una restricción de unicidad o una restricción de comprobación de tabla.
nombre-correlación	Identificador que designa una tabla resultante.
nombre-cursor	Identificador que designa un cursor SQL. Para compatibilidad entre sistemas principales, se puede utilizar un carácter de guión en el nombre.
nombre-fuente-datos	Identificador que designa una fuente de datos. Este identificador es el primero de las tres partes del nombre-objeto-remoto.
nombre-descriptor	Dos puntos seguidos de un identificador del lenguaje principal que designa un área de descriptores SQL (SQLDA). Consulte el apartado “Referencias a variables del lenguaje principal” en la página 151 para ver una descripción de un identificador del lenguaje principal. Observe que un nombre-descriptor nunca incluye una variable indicadora.
nombre-tipo-diferenciado	Nombre calificado o no calificado que designa un nombre-tipo-diferenciado. Un nombre-tipo-diferenciado no calificado de una sentencia de SQL está calificado implícitamente por el gestor de bases de datos según el contexto.
nombre-supervisor-sucesos	Identificador que designa un supervisor de sucesos.
nombre-correlación-funciones	Identificador que designa una correlación de funciones.
nombre-función	Nombre calificado o no calificado que designa una función. Un nombre-función no calificado

de una sentencia de SQL está calificado implícitamente por el gestor de bases de datos, según el contexto.

nombre-grupo	Identificador no calificado que designa un grupo de transformación definido para un tipo estructurado.
variable-lengprinc	Secuencia de símbolos que designa una variable del lenguaje principal. Una variable del lenguaje principal incluye, como mínimo, un identificador del lenguaje principal, tal como se explica en el apartado “Referencias a variables del lenguaje principal” en la página 151.
nombre-índice	Nombre calificado con esquema que designa un índice o una especificación de índice.
etiqueta	Identificador que designa una etiqueta en un procedimiento SQL.
nombre-método	Identificador que designa un método. El contexto de esquema de un método está determinado por el esquema del tipo indicado (o de un supertipo del tipo indicado) del método.
apodo	Nombre, calificado por un esquema, que designa una referencia de servidor federado a una tabla o vista.
nombre-gruponodos	Identificador que designa un grupo de nodos.
nombre-paquete	Nombre calificado con esquema que designa un paquete.
nombre-parámetro	Identificador que designa un parámetro al que se puede hacer referencia en un procedimiento, función definida por el usuario, método o extensión de índice.
nombre-procedimiento	Nombre calificado o no calificado que designa un procedimiento. Un nombre-procedimiento no calificado en una sentencia de SQL está calificado implícitamente por el gestor de bases de datos según el contexto.
nombre-autorización-remoto	Identificador que designa un usuario de

Convenios de denominación

	fuente de datos. Las normas para los nombres de autorización varían de fuente de datos a fuente de datos.
nombre-función-remota	Nombre que designa una función registrada en una base de datos de fuente de datos.
nombre-objeto-remoto	Nombre de tres partes que designa una tabla de fuente de datos o vista y que identifica la fuente de datos donde reside la tabla o la vista. Las partes de este nombres son nombre-fuente-datos, nombre-esquema-remoto y nombre-tabla-remota.
nombre-esquema-remoto	Nombre que designa el esquema al que pertenece una tabla de fuente de datos o vista. Este nombre es el segundo de las tres partes del nombre-objeto-remoto.
nombre-tabla-remota	Nombre que designa una tabla o una vista en una fuente de datos. Este nombre es el tercero de las tres partes del nombre-objeto-remoto.
nombre-tipo-remoto	Tipo de datos soportado por una base de datos de fuente de datos. No utilice el formato largo para los tipos integrados en el sistema (por ejemplo, utilice CHAR en vez de CHARACTER).
nombre-puntosalvar	Identificador que designa un punto de salvar.
nombre-esquema	Identificador que proporciona una agrupación lógica de objetos SQL. Un nombre-esquema que se utiliza como calificador del nombre de un objeto puede establecerse implícitamente: <ul style="list-style-type: none">• a partir del valor del registro especial CURRENT SCHEMA• a partir del valor de la opción de precompilación/enlace lógico QUALIFIER• sobre la base de un algoritmo de resolución que utilice el registro especial CURRENT PATH• sobre la base del nombre de esquema de otro objeto en la misma sentencia de SQL.

Para evitar complicaciones, es recomendable no utilizar "SESSION" como nombre de esquema, excepto para el esquema

	correspondiente a tablas temporales globales (las cuales deben utilizar el nombre de esquema "SESSION").
nombre-servidor	Identificador que designa un servidor de aplicaciones. En un sistema federado, nombre-servidor también designa el nombre local de una fuente de datos.
nombre-específico	Nombre, calificado o no calificado, que designa un nombre específico. Un nombre-específico no calificado en una sentencia de SQL está calificado implícitamente por el gestor de bases de datos según el contexto.
nombre-variable-SQL	Define el nombre de una variable local en una sentencia de un procedimiento SQL. Los nombres de variables SQL se pueden utilizar en otras sentencias de SQL donde esté permitido un nombre de variable del lenguaje principal. El nombre puede estar calificado por la etiqueta de la sentencia compuesta donde se declaró la variable SQL.
nombre-sentencia	Identificador que designa una sentencia de SQL preparada.
nombre-supertipo	Nombre calificado o no calificado que designa el supertipo de un nombre-tipo. Un nombre-supertipo no calificado en una sentencia de SQL está calificado implícitamente por el gestor de bases de datos según el contexto.
nombre-tabla	Nombre calificado con esquema que designa una tabla.
nombre-espacio-tablas	Identificador que designa un espacio de tablas.
nombre-desencadenante	Nombre, calificado por un esquema, que designa un desencadenante.
nombre-correlación-tipos	Identificador que designa una correlación de tipos de datos.
nombre-tipo	Nombre calificado o no calificado que designa un nombre-tipo. Un nombre-tipo no calificado

Convenios de denominación

	en una sentencia de SQL está calificado implícitamente por el gestor de bases de datos según el contexto.
nombre-tabla-tipo	Nombre, calificado por un esquema, que designa una tabla con tipo.
nombre-vista-tipo	Nombre calificado con esquema que designa una vista con tipo.
nombre-vista	Nombre calificado con esquema que designa una vista.
nombre-reiniciador	Identificador que designa un reiniciador.

Seudónimos

Unseudónimo de tabla se puede considerar como un nombre alternativo de una tabla o una vista. Por lo tanto, en una sentencia de SQL se puede hacer referencia a una tabla o a una vista por su nombre o por suseudónimo de tabla.

Unseudónimo se puede utilizar siempre que se pueda utilizar un nombre de tabla o vista. Se puede crear unseudónimo aunque no exista el objeto (aunque debe existir en el momento de compilar una sentencia que hace referencia al mismo). Puede hacer referencia a otroseudónimo si no se realizan referencias circulares ni repetitivas a lo largo de la cadena deseudónimos. Unseudónimo sólo puede hacer referencia a una tabla, una vista o unseudónimo de la misma base de datos. Unseudónimo no se puede utilizar cuando se espera un nombre de tabla nueva o de vista nueva como, por ejemplo, en las sentencias CREATE TABLE o CREATE VIEW; por ejemplo, si se crea elseudónimo PERSONAL, una sentencia posterior como, por ejemplo, CREATE TABLE PERSONAL... generaría un error.

La opción de para hacer referencia a una tabla o vista mediante unseudónimo no se muestra explícitamente en los diagramas de sintaxis ni se menciona en la descripción de la sentencia de SQL.

Unseudónimo no calificado nuevo no puede tener el mismo nombre completamente calificado que una tabla, una vista o unseudónimo existente.

El efecto de utilizar unseudónimo en una sentencia de SQL es similar al de la sustitución de texto. Elseudónimo, que debe estar definido cuando se compila la sentencia de SQL, se sustituye en el momento de la compilación de la sentencia por el nombre base calificado de la tabla o vista. Por ejemplo, si PBIRD.SALES es unseudónimo para DSPN014.DIST4_SALES_148, entonces en el momento de la compilación:

```
SELECT * FROM PBIRD.SALES
```

se convierte en realidad en

```
SELECT * FROM DSPN014.DIST4_SALES_148
```

En un sistema federado, los usos y restricciones de la sentencia mencionada no sólo se aplican a los seudónimo de tabla, sino también a los seudónimo de apodos. Por consiguiente, un seudónimo de apodo se puede utilizar en lugar del apodo en una sentencia de SQL; se puede crear un seudónimo para un apodo que aún no exista, siempre que el apodo se cree antes de que las sentencias que hacen a la referencia se compilen; un seudónimo para un apodo puede hacer referencia a otro seudónimo para este apodo y así sucesivamente.

En la tolerancia de sintaxis de otras aplicaciones del sistema de gestión de bases de datos relacionales, se puede utilizar SYNONYM en vez de ALIAS en las sentencias CREATE ALIAS y DROP ALIAS.

Consulte “CREATE ALIAS” en la página 670 para obtener más información sobre los seudónimos.

ID de autorización y nombres-autorización

Un *ID de autorización* es una serie de caracteres obtenida por el gestor de bases de datos cuando se establece una conexión entre el gestor de bases de datos y un proceso de aplicación o un proceso de preparación de programa. Designa un conjunto de privilegios. También puede designar a un usuario o a un grupo de usuarios, pero su propiedad no la controla el gestor de bases de datos.

El gestor de bases de datos utiliza los ID de autorización para proporcionar:

- El control de autorizaciones de sentencias de SQL
- El valor por omisión para la opción de precompilación/enlace lógico QUALIFIER y el registro especial CURRENT SCHEMA. Además, se incluye el ID de autorización en la opción de precompilación/enlace lógico FUNCPATH y el registro especial CURRENT PATH por omisión.

Se aplica un ID de autorización a cada sentencia de SQL. El ID de autorización que se aplica a una sentencia de SQL estática es el ID de autorización que se utiliza durante el enlace de programas. El ID de autorización correspondiente a una sentencia de SQL dinámica se basa en la opción DYNAMICRULES proporcionada durante el proceso de enlace para el paquete que emite la sentencia de SQL dinámica. Para un paquete enlazado con DYNAMICRULES RUN, el ID de autorización utilizado es el ID de autorización del usuario que ejecuta el paquete. Para un paquete enlazado con

ID de autorización y nombres-autorización

DYNAMICRULES BIND, el ID de autorización utilizado es el ID de autorización del paquete. Este ID se denomina *ID de autorización de ejecución*.

Un *nombre-autorización* especificado en una sentencia de SQL no se debe confundir con el ID de autorización de la sentencia. Un nombre-autorización es un identificador que se utiliza en varias sentencias de SQL. Un nombre-autorización se utiliza en una sentencia CREATE SCHEMA para designar al propietario del esquema. El nombre de autorización se utiliza en las sentencias GRANT y REVOKE para designar el destino de la operación de otorgamiento (grant) o revocación (revoke). Observe que la premisa de un otorgamiento de privilegios a X es que X, o un miembro del grupo X, será posteriormente el ID de autorización de las sentencias que necesiten dichos privilegios.

Ejemplos:

- Suponga que SMITH es el id de usuario del ID de autorización que el gestor de bases de datos ha obtenido al establecer la conexión con el proceso de aplicación. La siguiente sentencia se ejecuta interactivamente:

```
GRANT SELECT ON DEPTT TO KEENE
```

SMITH es el ID de autorización de la sentencia. Por lo tanto, el valor por omisión del registro especial CURRENT SCHEMA en una sentencia de SQL dinámica y el valor por omisión de la opción de precompilación/enlace QUALIFIER en el SQL estático es SMITH. De este modo, la autorización para ejecutar la sentencia se compara con SMITH y SMITH es el calificador implícito de *nombre-tabla*, de acuerdo con las reglas de calificación descritas en “Convenios de denominación y calificaciones implícitas de nombres de objetos” en la página 72.

KEENE es un nombre-autorización especificado en la sentencia. Se otorga el privilegio SELECT en SMITH.TDEPT a KEENE.

- Suponga que SMITH tiene autorización de administración y es el ID de autorización de las siguientes sentencias de SQL dinámico sin que se emita ninguna sentencia SET SCHEMA durante la sesión:

```
DROP TABLE TDEPT
```

Elimina la tabla SMITH.TDEPT.

```
DROP TABLE SMITH.TDEPT
```

Elimina la tabla SMITH.TDEPT.

```
DROP TABLE KEENE.TDEPT
```

Elimina la tabla KEENE.TDEPT. Observe que KEENE.TDEPT y SMITH.TDEPT son tablas diferentes.

```
CREATE SCHEMA PAYROLL AUTHORIZATION KEENE
```

KEENE es el nombre-autorización especificado en la sentencia que crea un esquema denominado PAYROLL. KEENE es el propietario del esquema PAYROLL y se le otorgan los privilegios CREATEIN, ALTERIN y DROPIN con la posibilidad de otorgarlos a otros.

Características del SQL dinámico en la ejecución

La opción OWNER de los mandatos BIND y PRECOMPILE define el ID de autorización del paquete.

La opción QUALIFIER de los mandatos BIND y PRECOMPILE define el calificador implícito para los objetos no calificados contenidos en el paquete.

La opción DYNAMICRULES de los mandatos BIND y PRECOMPILE determina si las sentencias del SQL dinámico se procesan en la ejecución con las reglas de la ejecución, DYNAMICRULES RUN, o con las reglas de la vinculación, DYNAMICRULES BIND. Estas normas indican el valor utilizado como ID de autorización y el valor utilizado para la calificación implícita de referencias a objeto no calificado en sentencias del SQL dinámico. Las opciones DYNAMICRULES tienen los efectos descritos en las tablas siguientes:

Tabla 1. Características del SQL estático a las que afectan OWNER y QUALIFIER

Característica	Si sólo se especifica OWNER	Si sólo se especifica QUALIFIER	Si se especifican QUALIFIER y OWNER
ID de autorización utilizado	ID de usuario especificado en la opción OWNER del mandato BIND	ID de usuario que vincula el paquete	ID de usuario especificado en la opción OWNER del mandato BIND
Valor utilizado de calificación de objeto no calificado	ID de usuario especificado en la opción OWNER del mandato BIND	ID de usuario especificado en la opción QUALIFIER del mandato BIND	ID de usuario especificado en la opción QUALIFIER del mandato BIND

Tabla 2. Características del SQL dinámico a las que afectan DYNAMICRULES, OWNER y QUALIFIER

Característica	RUN	BIND
ID de autorización utilizado	ID de usuario que ejecuta el paquete	ID de autorización para el paquete
Valor utilizado de calificación de objeto no calificado	Registro especial CURRENT SCHEMA	ID de autorización para el paquete

Consideraciones relativas a la opción DYNAMICRULES:

Características del SQL dinámico en la ejecución

- No se utilizará el registro especial CURRENT SCHEMA para calificar las referencias a objeto no calificado en las sentencias del SQL dinámico ejecutadas desde un paquete vinculado con DYNAMICRULES BIND. En su lugar, DB2 utilizará el calificador por omisión del paquete tal como se muestra en la tabla. Esto es así incluso después de emitir la sentencia SET CURRENT SCHEMA para cambiar el registro especial CURRENT SCHEMA; el valor del registro se cambiará pero no se utilizará.
- No se pueden utilizar las siguientes sentencias del SQL preparado dinámicamente de un paquete vinculado con la opción DYNAMICRULES BIND: GRANT, REVOKE, ALTER, CREATE, DROP, COMMENT ON, RENAME, SET INTEGRITY, SET EVENT MONITOR STATE y consultas que hacen referencia a un apodo.
- Si se hace referencia a múltiples paquetes durante una sola conexión, el SQL dinámico se comportará de acuerdo a las opciones BIND para el paquete en el que se vincula una sentencia.
- Es importante tener presente que cuando se vincula un paquete con DYNAMICRULES BIND, el vinculador del paquete no debe haberles otorgado ninguna autorización que no desee que el usuario del paquete tenga, ya que una sentencia dinámica utilizará el ID de autorización del propietario del paquete.

Los ID de autorización y la preparación de sentencias

Si se especifica VALIDATE BIND al ejecutar BIND, se deben tener los privilegios necesarios para manejar tablas y vistas. Si los privilegios u objetos referenciados no existen y está en vigor la opción SQLERROR NOPACKAGE, la operación de enlace no es efectiva. Si se especifica SQLERROR CONTINUE, el enlace se realiza satisfactoriamente y se marcan las sentencias erróneas. Si se intenta ejecutar una sentencia marcada como errónea, se producirá un error en la aplicación.

Si un paquete se enlaza utilizando VALIDATE RUN, se ejecuta todo el proceso normal de enlace, pero no es necesario que en ese momento existan los privilegios necesarios para utilizar las tablas y vistas referenciadas en la aplicación. Si durante el enlace no existe algún privilegio necesario para una sentencia, se realiza un enlace parcial la primera vez que se ejecuta la sentencia en una aplicación, y todos los privilegios necesarios para la sentencia deben existir. Si falta algún privilegio, la ejecución de la sentencia no es efectiva. La comprobación de autorizaciones que se realiza durante la ejecución utiliza el ID de autorización del propietario del paquete.

Tipos de datos

Para obtener información sobre la especificación de los tipos de datos de las columnas, consulte el apartado “CREATE TABLE” en la página 835.

La unidad más pequeña de datos que se puede manipular en SQL se denomina un *valor*. La forma en que se interpretan los valores depende del tipo de datos de su fuente. Las fuentes de los valores son:

- Constantes
- Columnas
- Variables del lenguaje principal
- Funciones
- Expresiones
- Registros especiales.

DB2 da soporte a varios tipos de datos incorporados, que se describen en esta sección. También proporciona soporte para los tipos de datos definidos por el usuario. Consulte el apartado “Tipos definidos por el usuario” en la página 96 para ver una descripción de los tipos de datos definidos por el usuario.

La Figura 10 en la página 84 ilustra los tipos de datos incorporados soportados.

Tipos de datos

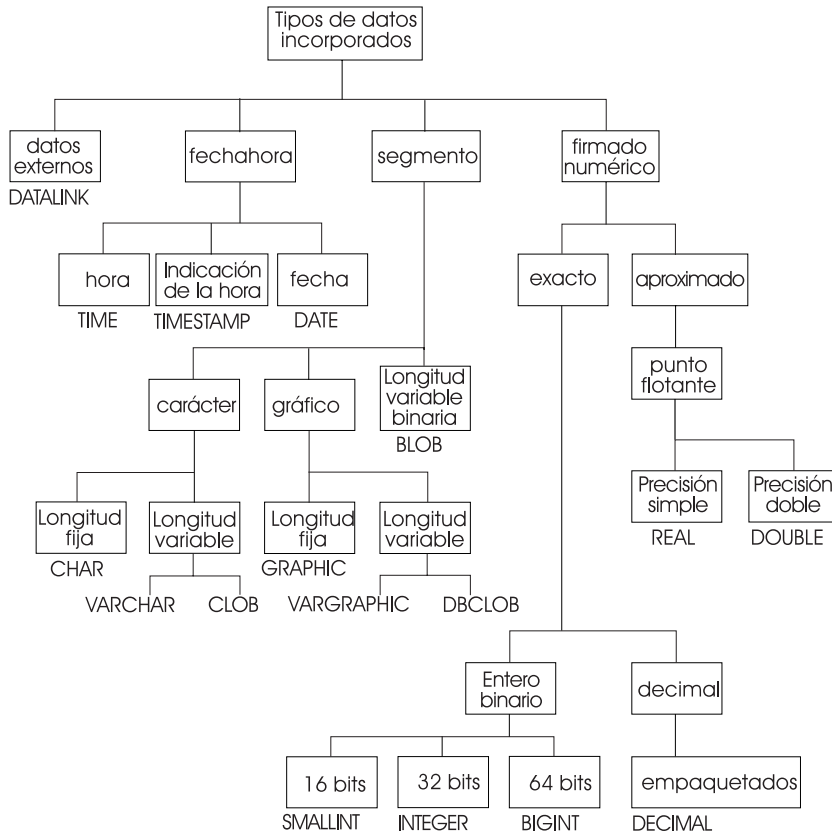


Figura 10. Tipos de datos incorporados soportados

Nulos

Todos los tipos de datos incluyen el valor nulo. El valor nulo es un valor especial que se diferencia de todos los valores que no son nulos y, por lo tanto, indica la ausencia de un valor (no nulo). Aunque todos los tipos de datos incluyen el valor nulo, las columnas definidas como NOT NULL no pueden contener valores nulos.

Gran objeto (LOB)

El término *gran objeto* y su acrónimo genérico *LOB* se utilizan para hacer referencia a cualquier tipo de datos BLOB, CLOB o DBCLOB. Los valores LOB están sujetos a las restricciones que se aplican a los valores LONG VARCHAR tal como se especifica en el apartado “Restricciones aplicables a la utilización de series de caracteres de longitud variable” en la página 87. En series LOB, estas restricciones se aplican incluso cuando el atributo de longitud de la serie es de 254 bytes o menos.

Series de gran objeto de caracteres (CLOB)

Un *Gran objeto de caracteres (CLOB)* es una serie de longitud variable medida en bytes que puede tener un máximo de 2 gigabytes (2 147 483 647 bytes) de longitud. Un *CLOB* se utiliza para almacenar grandes datos basados en caracteres SBCS o mixtos (SBCS y MBCS) como, por ejemplo, documentos escritos con un solo juego de caracteres (y, por lo tanto, tienen una página de códigos SBCS o mixta asociada a ellos). Observe que un *CLOB* se considera una serie de caracteres.

Series de gran objeto de caracteres de doble byte (DBCLOB)

Un *Gran objeto de caracteres de doble byte (DBCLOB)* es una serie de longitud variable de caracteres de doble byte que puede tener hasta 1.073.741.823 caracteres de longitud. Un *DBCLOB* se utiliza para almacenar grandes datos basados en caracteres DBCS como, por ejemplo, documentos escritos con un solo juego de caracteres (y, por lo tanto, tienen un CCSID DBCS asociado). Observe que un *DBCLOB* se considera una serie gráfica.

Gran objeto binario (BLOB)

Un *Gran objeto binario (BLOB)* es una serie de longitud variable medida en bytes que puede tener un máximo de 2 gigabytes (2 147 483 647 bytes) de longitud. Un *BLOB* está pensado principalmente para contener datos que no son tradicionales como, por ejemplo, imágenes, voz y de soportes mixtos. Otra utilización es contener datos estructurados para que las funciones definidas por el usuario y los tipos definidos por el usuario los exploten. Igual que las series de caracteres FOR BIT DATA, las series *BLOB* no están asociadas con ningún juego de caracteres.

Manipulación de grandes objetos (LOB) con localizadores

Puesto que los valores *LOB* son muy grandes, la transferencia de dichos valores del servidor de bases de datos a las variables del lenguaje principal del programa de aplicación cliente puede tardar mucho tiempo. Sin embargo, también es cierto que, normalmente, los programas de aplicación procesan valores *LOB*, de fragmento en fragmento, en lugar de como un todo. Para los casos en que la aplicación no necesita (o no desea) que todo el valor *LOB* se almacene en la memoria de la aplicación, la aplicación puede hacer referencia a un valor *LOB* a través de un localizador de gran objeto (localizador de *LOB*).

Un *localizador de gran objeto* o localizador de *LOB* es una variable del lenguaje principal con un valor que representa un solo valor *LOB* en el servidor de bases de datos. Los localizadores de *LOB* se han desarrollado para proporcionar a los usuarios un mecanismo mediante el cual puedan manipular fácilmente objetos muy grandes en programas de aplicación sin necesidad de almacenar todo el valor *LOB* en la máquina cliente en la que se ejecuta el programa de aplicación.

Tipos de datos

Por ejemplo, cuando se selecciona un valor LOB, un programa de aplicación podría seleccionar todo el valor LOB y colocarlo en una variable del lenguaje principal igualmente grande (que se puede aceptar si el programa de aplicación va a procesar todo el valor LOB a la vez), o podría seleccionar el valor LOB del localizador de LOB en su lugar. Mediante la utilización del localizador de LOB, el programa de aplicación puede emitir operaciones de bases de datos posteriores en el valor LOB (por ejemplo, la aplicación de las funciones escalares SUBSTR, CONCAT, VALUE, LENGTH, la realización de una asignación, la búsqueda del LOB con LIKE o POSSTR, o la aplicación de UDF en el LOB) suministrando el valor del localizador como entrada. La salida resultante de la operación del localizador, por ejemplo, la cantidad de datos asignados a una variable del lenguaje principal cliente, sería normalmente un subconjunto pequeño del valor LOB de entrada.

Los localizadores de LOB también pueden representar, además de valores base, el valor asociado con una expresión LOB. Por ejemplo, un localizador de LOB puede representar el valor asociado con:

```
SUBSTR( <lob 1> CONCAT <lob 2> CONCAT <lob 3>, <inicio>, <longitud> )
```

En variables del lenguaje principal normales de un programa de aplicación, cuando se selecciona un valor nulo en dicha variable del lenguaje principal, la variable indicadora se establece en -1, lo que significa que el valor es nulo. Sin embargo, en el caso de los localizadores de LOB, el significado de las variables indicadoras es levemente diferente. Puesto que una variable del lenguaje principal del localizador en sí nunca puede ser nula, un valor negativo de variable indicadora significa que el valor LOB representado por el localizador de LOB es nulo. La información de nulo se mantiene local para el cliente en virtud del valor de la variable indicadora — el servidor no hace ningún seguimiento de los valores nulos con localizadores válidos.

Es importante comprender que un localizador de LOB representa un valor, no una fila ni una ubicación en la base de datos. Cuando se ha seleccionado un valor en un localizador, no hay ninguna operación que se pueda efectuar en la fila o tabla originales que afecte al valor al que hace referencia el localizador. El valor asociado con un localizador es válido hasta que finaliza la transacción o hasta que el localizador se libera explícitamente, lo primero que se produzca. Los localizadores no fuerzan copias adicionales de los datos para que puedan realizar su función. En su lugar, el mecanismo del localizador almacena una descripción del valor LOB base. La materialización del valor LOB (o expresión, tal como se muestra arriba) se difiere hasta que se asigna realmente a alguna ubicación — en un almacenamiento intermedio del usuario en la forma de una variable del lenguaje principal o en otro valor del campo del registro de la base de datos.

Un localizador de LOB es sólo un mecanismo utilizado para hacer referencia a un valor LOB durante una transacción; no persiste más allá de la transacción

en la que se ha creado. Tampoco es un tipo de base de datos; nunca se almacena en la base de datos y, como resultado, no puede participar en vistas ni en restricciones de comprobación. Sin embargo, puesto que un localizador es una representación cliente de un tipo LOB, hay `SQLTYPE` para localizadores de LOB para que puedan describirse dentro de una estructura `SQLDA` que se utiliza por sentencias `FETCH`, `OPEN` y `EXECUTE`.

Series de caracteres

Una *serie de caracteres* es una secuencia de bytes. La longitud de la serie es el número de bytes en la secuencia. Si la longitud es cero, el valor se denomina la *serie vacía*. Este valor no debe confundirse con el valor nulo.

Series de caracteres de longitud fija

Todos los valores de una columna de series de longitud fija tienen la misma longitud, que está determinada por el atributo de longitud de la columna. El atributo de longitud debe estar entre 1 y 254, inclusive.

Series de caracteres de longitud variable

Las series de caracteres de longitud variable son de tres tipos: `VARCHAR`, `LONG VARCHAR` y `CLOB`.

- Los tipos de `VARCHAR` son series de longitud variable de un máximo de 32 672 bytes.
- Los tipos `LONG VARCHAR` son series de longitud variable de un máximo de 32.700 bytes.
- Los tipos `CLOB` son series de longitud variable de un máximo de 2 gigabytes.

Restricciones aplicables a la utilización de series de caracteres de longitud variable: Se aplican restricciones especiales a una expresión que dé como resultado datos de tipo serie, de longitud variable, cuya longitud máxima sea mayor que 255 bytes; dichas expresiones no están permitidas en:

- Una lista `SELECT` precedida por `DISTINCT`
- Una cláusula `GROUP BY`
- Una cláusula `ORDER BY`
- Una función de columna con `DISTINCT`
- Una subselección de un operador de conjunto que no sea `UNION ALL`.

Además de las restricciones descritas anteriormente, las expresiones que den como resultado datos de tipo `LONG VARCHAR` o `CLOB` no están permitidas en:

- Un predicado Básico, Cuantificado, `BETWEEN` o `IN`
- Una función de columna
- Las funciones escalares `VARGRAPHIC`, `TRANSLATE` y de indicación de fecha y hora

Tipos de datos

- El operando patrón de un predicado LIKE o el operando de serie de búsqueda de una función POSSTR
- La representación en forma de serie de caracteres de un valor de indicación de fecha y hora

Las funciones del esquema SYSFUN que toman como argumento VARCHAR no aceptarán las VARCHAR que tengan más de 4.000 bytes de longitud como argumento. Sin embargo, muchas de estas funciones también pueden tener una signature alternativa que acepte un CLOB(1M). Para estas funciones, el usuario puede convertir explícitamente las series VARCHAR mayores que 4.000 en datos CLOB y luego reconvertir el resultado en datos VARCHAR de la longitud deseada.

Series de caracteres terminadas en nulo

Las series de caracteres terminadas en nulo que se encuentran en C se manejan de manera diferente, dependiendo del nivel de estándares de la opción de precompilación. Consulte la sección específica del lenguaje C en el manual *Application Development Guide* para obtener más información sobre el tratamiento de las series de caracteres terminadas en nulo.

Subtipos de caracteres

Cada serie de caracteres se define con más detalle como:

- Datos de bits** Datos que no están asociado con una página de códigos.
- Datos SBCS** Datos en los que cada carácter está representado por un solo byte.
- Datos mixtos** Datos que pueden contener una mezcla de caracteres de un juego de caracteres de un solo byte (SBCS) y de un juego de caracteres de múltiples bytes (MBCS).

Consideraciones acerca de SBCS y MBCS: Los datos SBCS sólo están soportados en una base de datos SBCS. Sólo se da soporte a los datos mixtos en una base de datos MBCS.

Series gráficas

Una *serie gráfica* es una secuencia de bytes que representa datos de caracteres de doble byte. La longitud de la serie es el número de caracteres de doble byte de la secuencia. Si la longitud es cero, el valor se denomina la serie vacía. Este valor no debe confundirse con el valor nulo.

Las series gráficas no se validan para asegurarse de que sus valores sólo contienen elementos de código de caracteres de doble byte.¹⁰ Mejor dicho, el

10. La excepción a esta regla es una aplicación precompilada con la opción WCHARTYPE CONVERT. En este caso, sí que se efectúa la validación. Vea la sección "Programming in C and C++" del manual *Application Development Guide* para obtener detalles.

gestor de bases de datos supone que los datos de caracteres de doble byte están contenidos dentro de campos de datos gráficos. El gestor de bases de datos comprueba que un valor de serie gráfica tenga como longitud un número par de bytes.

Un tipo de datos de serie gráfica puede tener una longitud fija o variable; la semántica de la longitud fija y variable es análoga a las definidas para los tipos de datos de series de caracteres.

Series gráficas de longitud fija

Todos los valores de una columna de series gráficas de longitud fija tienen la misma longitud, que viene determinada por el atributo de longitud de la columna. El atributo de longitud debe estar entre 1 y 127, inclusive.

Series gráficas de longitud variable

Las series gráficas de longitud variable son de tres tipos: VARGRAPHIC, LONG VARGRAPHIC y DBCLOB.

- Los tipos VARGRAPHIC son series de longitud variable de hasta un máximo de 16.336 caracteres de doble byte.
- Los tipos LONG VARGRAPHIC son series de longitud variable de un máximo de 16.350 caracteres de doble byte.
- Los tipos DBCLOB son series de longitud variable de un máximo de 1 073 741 823 caracteres de doble byte.

Se aplican restricciones especiales a una expresión cuyo resultado sea un tipo de datos de serie gráfica de longitud variable cuya longitud máxima sea mayor que 127 bytes. Dichas restricciones son las mismas que las especificadas en el apartado “Restricciones aplicables a la utilización de series de caracteres de longitud variable” en la página 87.

Series gráficas terminadas en nulo

Las series gráficas terminadas en nulo que se encuentran en C se manejan de manera diferente, dependiendo del nivel de estándares de la opción de precompilación. Consulte la sección específica al lenguaje C en el manual *Application Development Guide* para obtener más información sobre el tratamiento de series gráficas de terminación nula.

Este tipo de datos no puede crearse en una tabla. Sólo se puede utilizar para insertar datos en la base de datos y recuperarlos de la misma.

Serie binaria

Una *serie binaria* es una secuencia de bytes. A diferencia de la serie de caracteres, que normalmente contiene datos de texto, una serie binaria se utiliza para contener datos que no son tradicionales como, por ejemplo, imágenes. Observe que las series de caracteres del subtipo ‘datos de bits’ pueden utilizarse para fines similares, pero los dos tipos de datos no son

Tipos de datos

compatibles. La función escalar BLOB puede utilizarse para convertir un carácter, de serie de bits, en una serie binaria. La longitud de una serie binaria es el número de bytes. No está asociada con una página de códigos. Las series binarias tienen las mismas restricciones que las series de caracteres (consulte los detalles en el apartado “Restricciones aplicables a la utilización de series de caracteres de longitud variable” en la página 87).

Números

Todos los números tienen un signo y una precisión. La *precisión* es el número de bits o de dígitos excluyendo el signo. El signo se considera positivo si el valor de un número es cero.

Entero pequeño (SMALLINT)

Un *entero pequeño* es un entero de dos bytes con una precisión de 5 dígitos. El rango de pequeños enteros va de -32 768 a 32 767.

Entero grande (INTEGER)

Un *entero grande* es un entero de cuatro bytes con una precisión de 10 dígitos. El rango de enteros grandes va de -2 147 483 648 a +2 147 483 647.

Entero superior (BIGINT)

Un *entero superior* es un entero de ocho bytes con una precisión de 19 dígitos. El rango de enteros grandes va de -9 223 372 036 854 775 808 a +9 223 372 036 854 775 807.

Coma flotante de precisión simple (REAL)

Un número de *coma flotante de precisión simple* es una aproximación de 32 bits de un número real. El número puede ser cero o puede estar en el rango de -3,402E+38 a -1,175E-37, o de 1,175E-37 a 3,402E+38.

Coma flotante de doble precisión (DOUBLE o FLOAT)

Una número de *coma flotante de doble precisión* es una aproximación de 64 bits de un número real. El número puede ser cero o puede estar en el rango de -1,79769E+308 a -2,225E-307, o de 2,225E-307 a 1,79769E+308.

Decimal (DECIMAL o NUMERIC)

Un valor *decimal* es un número decimal empaquetado con una coma decimal implícita. La posición de la coma decimal la determinan la precisión y la escala del número. La escala, que es el número de dígitos en la parte de la fracción del número, no puede ser negativa ni mayor que la precisión. La precisión máxima es de 31 dígitos. Para obtener información acerca de la representación decimal empaquetada, consulte el apartado “Números decimales empaquetados” en la página 1279.

Todos los valores de una columna decimal tienen la misma precisión y escala. El rango de una variable decimal o de los números de una columna decimal

es de $-n$ a $+n$, donde el valor absoluto de n es el número mayor que puede representarse con la precisión y escalas aplicables. El rango máximo es de $-10^{31}+1$ a $10^{31}-1$.

Valores de indicación de fecha y hora

Los tipos de datos de indicación de fecha y hora se describen a continuación. Aunque los valores de indicación de fecha y hora se pueden utilizar en algunas operaciones aritméticas y de series y son compatibles con algunas series, no son ni series ni números.

Fecha

Una *fecha* es un valor que se divide en tres partes (año, mes y día). El rango de la parte correspondiente al año es de 0001 a 9999. El rango de la parte correspondiente al mes es de 1 a 12. El rango de la parte correspondiente al día es de 1 a x , donde x depende del mes.

La representación interna de una fecha es una serie de 4 bytes. Cada byte consta de 2 dígitos decimales empaquetados. Los 2 primeros bytes representan el año, el tercer byte el mes y el último byte el día.

La longitud de una columna DATE, tal como se describe en el SQLDA, es de 10 bytes, que es la longitud adecuada para una representación de serie de caracteres del valor.

Hora

Una *hora* es un valor que se divide en tres partes (hora, minuto y segundo) que indica una hora del día de un reloj de 24 horas. El rango de la parte correspondiente a la hora es de 0 a 24; mientras que el rango de las otras es de 0 a 59. Si la hora es 24, las especificaciones de los minutos y segundos será cero.

La representación interna de la hora es una serie de 3 bytes. Cada byte es 2 dígitos decimales empaquetados. El primer byte representa la hora, el segundo byte el minuto y el último byte el segundo.

La longitud de la columna TIME, tal como se describe en SQLDA, es de 8 bytes, que es la longitud adecuada para una representación de serie de caracteres del valor.

Indicación de la hora

Una *indicación de la hora* es un valor dividido en siete partes (año, mes, día, hora, minuto, segundo y microsegundo) que indica una fecha y una hora como las definidas más arriba, excepto en que la hora incluye la especificación fraccional de los microsegundos.

Tipos de datos

La representación interna de una indicación de la hora es una serie de 10 bytes, cada uno de los cuales consta de 2 dígitos decimales empaquetados. Los 4 primeros bytes representan la fecha, los 3 bytes siguientes la hora y los últimos 3 bytes los microsegundos.

La longitud de una columna `TIMESTAMP`, tal como se describe en el `SQLDA`, es de 26 bytes, que es la longitud adecuada para la representación de serie de caracteres del valor.

Representaciones de serie de valores de indicación de fecha y hora

Los valores cuyos tipos de datos son `DATE`, `TIME` o `TIMESTAMP` se representan en un formato interno que es transparente al usuario de SQL. Sin embargo, las fechas, horas e indicaciones de la hora pueden representarse también mediante series de caracteres y dichas representaciones conciernen directamente al usuario de SQL ya que no hay ninguna constante ni variable cuyo tipo de datos sea `DATE`, `TIME` o `TIMESTAMP`. Por lo tanto, para poder recuperar un valor de indicación de fecha y hora debe asignarse a una variable de serie de caracteres. Observe que la función `CHAR` puede utilizarse para cambiar el valor de indicación de fecha y hora a una representación de serie. Normalmente, la representación de serie de caracteres es el formato por omisión de los valores de indicación de fecha y hora asociados con el código de país de la base de datos, a menos que se alteren temporalmente por la especificación de la opción `DATETIME` cuando se precompila el programa o se enlaza con la base de datos.

No importa su longitud, una serie de gran objeto o `LONG VARCHAR` no puede utilizarse como la serie que representa un valor de indicación de fecha y hora; de lo contrario, se genera un error (`SQLSTATE 42884`).

Cuando se utiliza una representación de serie válida de un valor de indicación de fecha y hora en una operación con un valor de indicación de fecha y hora interno, la representación de serie se convierte al formato interno de la fecha, hora o indicación de la hora antes de realizar la operación. Las secciones siguientes definen las representaciones de serie válidas de los valores de indicación de fecha y hora.

Series de fecha

Una representación de tipo serie de una fecha es una serie de caracteres que empieza por un dígito y que tiene una longitud de 8 caracteres como mínimo. Pueden incluirse blancos de cola; pueden omitirse los ceros iniciales de las partes correspondientes al mes y al día.

Los formatos válidos de tipo serie para las fechas están listados en la Tabla 1. Cada formato se identifica por el nombre e incluye una abreviatura asociada y un ejemplo de su utilización.

Tabla 3. Formatos para las representaciones de serie de fechas

Nombre del formato	Abreviatura	Formato de fecha	Ejemplo
International Standards Organization	ISO	aaaa-mm-dd	1991-10-27
Estándar IBM USA	USA	mm/dd/aaaa	10/27/1991
Estándar IBM European	EUR	dd.mm.aaaa	27.10.1991
Japanese Industrial Standard Christian Era	JIS	aaaa-mm-dd	1991-10-27
Definido por el sitio (consulte <i>Administration Guide</i>)	LOC	Depende del código del país de la base de datos	—

Series de hora

Una representación de serie de una hora es una serie de caracteres que empieza por un dígito y que tiene una longitud de 4 caracteres como mínimo. Pueden incluirse blancos de cola; puede omitirse un cero inicial de la parte correspondiente a la hora y pueden omitirse por completo los segundos. Si se omiten los segundos, se supone una especificación implícita de 0 segundos. De este modo, 13:30 es equivalente a 13:30:00.

Los formatos de serie válidos para las horas se listan en la Tabla 4. Cada formato se identifica por el nombre e incluye una abreviatura asociada y un ejemplo de su utilización.

Tabla 4. Formatos para representaciones de serie de horas

Nombre del formato	Abreviatura	Formato de la hora	Ejemplo
International Standards Organization ²	ISO	hh.mm.ss	13.30.05
Estándar IBM USA	USA	hh:mm AM o PM	1:30 PM
Estándar IBM European	EUR	hh.mm.ss	13.30.05
Era Japanese Industrial Standard Christian	JIS	hh:mm:ss	13:30:05
Definido por el sitio (consulte <i>Administration Guide</i>)	LOC	Depende del código del país de la base de datos	—

Tipos de datos

Notas:

1. En los formatos ISO, EUR y JIS, .ss (o :ss) es opcional.
2. La organización International Standards Organization ha cambiado recientemente el formato de la hora, de modo que ahora es idéntica a la de la Japanese Industrial Standard Christian Era. Por lo tanto, utilice el formato JIS si una aplicación necesita el formato actual de International Standards Organization.
3. En el caso del formato de serie de hora USA, puede omitirse la especificación de los minutos, indicando una especificación implícita de 00 minutos. Por lo tanto 1 PM es equivalente a 1:00 PM.
4. En el formato de hora USA, la hora no debe ser mayor que 12 y no puede ser 0, excepto en el caso especial de las 00:00 AM. Hay un solo espacio antes de AM y PM. Si se utiliza el formato JIS del reloj de 24 horas, la correspondencia entre el formato USA y el reloj de 24 horas es la siguiente:
 - 12:01 AM a 12:59 AM corresponde a 00:01:00 a 00:59:00.
 - 01:00 AM a 11:59 AM corresponde a 01:00:00 a 11:59:00.
 - 12:00 PM (mediodía) a 11:59 PM corresponde a 12:00:00 a 23:59:00.
 - 12:00 AM (medianoche) corresponde a 24:00:00 y 00:00 AM (medianoche) corresponde a 00:00:00.

Series de indicación de la hora

Una representación de serie de una indicación de la hora es una serie de caracteres que empieza por un dígito y que tiene una longitud de 16 caracteres como mínimo. La representación de serie completa de una indicación de la hora tiene el formato *aaaa-mm-dd-hh.mm.ss.nnnnnnnn*. Se pueden incluir los blancos de cola. Pueden omitirse los ceros iniciales de las partes correspondientes al mes, día y hora de la indicación de la hora y se pueden truncar los microsegundos u omitirse por completo. Si se omite cualquier cero de cola en la parte correspondiente a los microsegundos, se asume la especificación implícita de 0 para los dígitos que faltan. Por lo tanto, 1991-3-2-8.30.00 es equivalente a 1991-03-02-08.30.00.000000.

Las sentencias de SQL también dan soporte a la representación de serie ODBC de una indicación de la hora como un valor de entrada solamente. La representación de serie ODBC de una indicación de la hora tiene el formato *aaaa-mm-dd hh:mm:ss.nnnnnnnn*. Consulte el manual *CLI Guide and Reference* para obtener más información acerca de ODBC.

Consideraciones acerca de MBCS

Las series de fecha, hora e indicación de la hora sólo deben contener caracteres y dígitos de un solo byte.

Valores DATALINK

Un valor DATALINK es un valor encapsulado que contiene una referencia lógica de la base de datos a un archivo almacenado fuera de la base de datos. Los atributos de este valor encapsulado son los siguientes:

tipo de enlace

El tipo de enlace soportado actualmente es 'URL' (Uniform Resource Locator).

ubicación de datos

Es la ubicación de un archivo enlazado a una referencia dentro de DB2, en forma de un URL. Para este URL se pueden utilizar estos nombres de esquema:

- HTTP
- FILE
- UNC
- DFS

Las demás partes del URL son:

- el nombre del servidor de archivos para los esquemas HTTP, FILE y UNC
- el nombre de la célula para el esquema DFS
- la vía de acceso completa dentro del servidor de archivos o célula

Vea “Apéndice P. Especificaciones BNF para los enlaces de datos” en la página 1521 para obtener más información sobre las especificaciones BNF exactas para DATALINK (BNF es el acrónimo de Backus Naur Form).

comentario

Hasta 254 bytes de información descriptiva. Está pensado para los usos específicos de una aplicación, como, por ejemplo, una identificación alternativa o más detallada de la ubicación de los datos.

Los caracteres en blanco iniciales y de cola se eliminan durante el análisis de los atributos de ubicación de datos en forma de URL. Además, los nombres de esquema ('http', 'file', 'unc', 'dfs') y de sistema principal no son sensibles las mayúsculas/minúsculas y se convierten siempre a mayúsculas para guardarlos en la base de datos. Cuando se recupera un valor DATALINK de una base de datos, se intercala un símbolo de acceso dentro del atributo de URL cuando es apropiado. Este símbolo se genera dinámicamente y no es una parte permanente del valor DATALINK almacenado en la base de datos. Para obtener más detalles, vea las funciones escalares asociadas a DATALINK, a partir de “DLCOMMENT” en la página 320.

Un valor DATALINK puede tener solamente un atributo de comentario y un atributo vacío de ubicación de datos. Incluso es posible que un valor de este

Tipos de datos

tipo se almacene en una columna, pero, naturalmente, no se enlazará ningún archivo a esta columna. La longitud total del comentario y el atributo de ubicación de datos de un valor DATALINK está actualmente limitado a 200 bytes.

Observe que los DATALINK no pueden intercambiarse con un servidor DRDA.

Es importante distinguir entre estas referencias DATALINK a archivos y las variables de referencia a archivos LOB descritas en la sección “Referencias a las variables del lenguaje principal BLOB, CLOB y DBCLOB” en la página 154. La semejanza es que ambas contienen una representación de un archivo. No obstante:

- Los valores DATALINK quedan retenidos en la base de datos y tanto los enlaces como los datos de los archivos enlazados pueden considerarse una ampliación natural de datos en la base de datos.
- Las variables de referencia a archivos existen temporalmente en el cliente y pueden considerarse una alternativa al almacenamiento intermedio de un programa principal.

Se proporcionan funciones escalares incorporadas para crear un valor DATALINK (DLVALUE) y para extraer los valores encapsulados de un valor DATALINK (DLCOMMENT, DLLINKTYPE, DLURLCOMPLETE, DLURLPATH, DLURLPATHONLY, DLURLSCHEME, DLURLSERVER).

Tipos definidos por el usuario

Tipos diferenciados

Un *tipo diferenciado* es un tipo de datos definido por el usuario que comparte su representación interna con un tipo existente (su tipo “fuente”), pero se considera un tipo independiente e incompatible para la mayoría de operaciones. Por ejemplo, se desea definir un tipo de imagen, un tipo de texto y un tipo de audio, todos ellos tienen semánticas bastante diferentes, pero utilizan el tipo de datos incorporado BLOB para su representación interna.

El siguiente ejemplo ilustra la creación de un tipo diferenciado denominado AUDIO:

```
CREATE DISTINCT TYPE AUDIO AS BLOB (1M)
```

Aunque AUDIO tenga la misma representación que el tipo de datos incorporado BLOB, se considera que es un tipo independiente que no se puede comparar a un BLOB ni a ningún otro tipo. Esto permite la creación de funciones escritas especialmente para AUDIO y asegura que dichas funciones no se aplicarán a ningún otro tipo (imágenes, texto, etc.).

Los tipos diferenciados se identifican por los identificadores calificados. Si no se utiliza el nombre de esquema para calificar el nombre del tipo diferenciado cuando se emplea en sentencias que no son CREATE DISTINCT TYPE, DROP DISTINCT TYPE o COMMENT ON DISTINCT TYPE, se busca en la *vía de acceso de SQL* por orden el primer esquema con un tipo diferenciado que coincida. La vía de acceso de SQL se describe en el apartado “CURRENT PATH” en la página 138.

Los tipos diferenciados dan soporte a una gran escritura asegurando que sólo aquellas funciones y operadores que están explícitamente definidos en un tipo diferenciado se puedan aplicar a sus instancias. Por esta razón, un tipo diferenciado no adquiere automáticamente las funciones y operadores de su tipo fuente, ya que estas podrían no tener ningún significado. (Por ejemplo, la función LENGTH del tipo AUDIO puede devolver la longitud de su objeto en segundos en lugar de bytes.)

Los tipos diferenciados que derivan de los tipos LONG VARCHAR, LONG VARGRAPHIC, LOB o DATALINK están sujetos a las mismas restricciones que su tipo fuente.

Sin embargo, se puede especificar explícitamente que algunas funciones y ciertos operadores se apliquen al tipo diferenciado definiendo funciones definidas por el usuario que derivan de funciones definidas en el tipo fuente del tipo diferenciado (vea “Comparaciones de tipos definidos por el usuario” en la página 119 para ver ejemplos). Los operadores de comparación se generan automáticamente para los tipos diferenciados definidos por el usuario, excepto los que utilicen LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB o DATALINK como tipo fuente. Además, se generan funciones para dar soporte a la difusión del tipo fuente en el tipo diferenciado y del tipo diferenciado en el tipo fuente.

Tipos estructurados

Un *tipo estructurado* es un tipo de datos definido por el usuario con una estructura definida en la base de datos. Contiene una secuencia de *atributos* con nombre, cada uno de los cuales tiene un tipo de datos. Un tipo estructurado también incluye un conjunto de especificaciones de método.

Un tipo estructurado puede utilizarse como tipo de una tabla, de una vista o de una columna. Cuando se utiliza como tipo para una tabla o vista, esa tabla o vista se denomina *tabla con tipo* o *vista con tipo*, respectivamente. Para las tablas con tipo y vistas con tipo, los nombres y tipos de datos de los atributos del tipo estructurado pasan a ser los nombres y tipos de datos de las columnas de esta tabla o vista con tipo. Las filas de la tabla o vista con tipo pueden considerarse una representación de instancias del tipo estructurado. Cuando se utiliza como tipo de datos para una columna, la columna contiene valores de ese tipo estructurado (o valores de cualquiera de los subtipos de

Tipos de datos

ese tipo, tal como se describe más abajo). Los métodos se utilizan para recuperar o manipular atributos de un objeto de columna estructurado.

Terminología: Un *supertipo* es un tipo estructurado para el que se han definido otros tipos estructurados, llamados *subtipos*. Un subtipo hereda todos los atributos y métodos de su supertipo y puede tener definidos otros atributos y métodos. El conjunto de tipos estructurados que están relacionados con un supertipo común se denomina *jerarquía de tipos* y el tipo que no tiene ningún supertipo se denomina el *tipo raíz* de la jerarquía de tipos.

El término subtipo se aplica a un tipo estructurado definido por el usuario y a todos los tipos estructurados definidos por el usuario que están debajo de él en la jerarquía de tipos. Por tanto, un subtipo de un tipo estructurado T es T y todos los tipos estructurados por debajo de T en la jerarquía. Un *subtipo propio* de un tipo estructurado T es un tipo estructurado por debajo de T en la jerarquía de tipos.

Existen restricciones respecto a la existencia de definiciones recursivas de tipos en una jerarquía de tipos. Por esta razón, es necesario desarrollar una forma abreviada de hacer referencia al tipo específico de definiciones recursivas que están permitidas. Se utilizan las definiciones siguientes:

- *Utiliza directamente*: Se dice que un tipo **A** utiliza directamente otro tipo **B** sólo si se cumple una de estas condiciones:
 1. el tipo **A** tiene un atributo del tipo **B**
 2. el tipo **B** es un subtipo de **A**, o un supertipo de **A**
- *Utiliza indirectamente*: Se dice que un tipo **A** utiliza indirectamente un tipo **B** sólo si se cumple una de estas condiciones:
 1. el tipo **A** utiliza directamente el tipo **B**
 2. el tipo **A** utiliza directamente cierto tipo **C**, y el tipo **C** utiliza indirectamente el tipo **B**

Un tipo puede no estar definido para que uno de sus tipos de atributo se utilice, directa o indirectamente, a sí mismo. Si es necesario tener una configuración así, considere la posibilidad de utilizar una referencia como atributo. Por ejemplo, en el caso de atributos de tipos estructurados, no puede existir una instancia de "empleado" que tenga el atributo "director" cuando "director" es de tipo "empleado". En cambio, puede existir un atributo "director" cuyo tipo sea REF(empleado).

Un tipo no se puede eliminar cuando ciertos otros objetos utilizan el tipo, ya sea directa o indirectamente. Por ejemplo, no se puede eliminar un tipo si una columna de una tabla o vista hace un uso directo o indirecto del tipo. Vea Tabla 29 en la página 1024 para conocer todos los objetos para los que pueden existir restricciones respecto a la eliminación de tipos.

Los valores de columna de tipo estructurado están sujetos a las restricciones que se aplican a los valores CLOB, tal como se especifica en “Restricciones aplicables a la utilización de series de caracteres de longitud variable” en la página 87.

Tipos de referencia (REF)

Un *tipo de referencia* es un tipo compañero de un tipo estructurado. De manera similar a un tipo diferenciado, un tipo de referencia es un tipo escalar que comparte una representación común con uno de los tipos de datos incorporados. Todos los tipos de la jerarquía de tipos comparten esta misma representación. La representación de un tipo de referencia se define cuando se crea el tipo raíz de una jerarquía de tipos. Cuando se utiliza un tipo de referencia, se especifica un tipo estructurado como parámetro del tipo. Este parámetro se denomina el *tipo de destino* de la referencia.

El destino de una referencia siempre es una fila de una tabla o vista con tipo. Cuando se utiliza un tipo de referencia, puede tener definido un *ámbito*. El ámbito identifica una tabla (denominada *tabla de destino*) o una vista (denominada *vista de destino*) que contiene la fila de destino de un valor de referencia. La tabla de destino o la vista de destino debe tener el mismo tipo que el tipo de destino del tipo de referencia. Una instancia de un tipo de referencia con ámbito identifica de forma exclusiva una fila en una tabla con tipo o en una vista con tipo, denominada *fila de destino*.

Promoción de los tipos de datos

Promoción de los tipos de datos

Los tipos de datos se pueden clasificar en grupos de tipos de datos relacionados. Dentro de estos grupos, existe un orden de prioridad en el que se considera que un tipo de datos precede a otro tipo de datos. Esta prioridad se utiliza para permitir la *promoción* de un tipo de datos a un tipo de datos posterior en el orden de prioridad. Por ejemplo, el tipo de datos CHAR puede promocionarse a VARCHAR, INTEGER puede promocionarse a DOUBLE-PRECISION pero CLOB NO es promocionable a VARCHAR.

La promoción de tipos de datos se utiliza en estos casos:

- cuando se efectúa una resolución de función (vea “Resolución de funciones” en la página 162)
- cuando se convierten tipos de datos definidos por el usuario (vea “Conversión entre tipos de datos” en la página 101)
- cuando se asignan tipos definidos por el usuario a tipos de datos incorporados (vea “Asignaciones de los tipos definidos por el usuario” en la página 113).

La Tabla 5 muestra la lista de prioridad (por orden) para cada tipo de datos y se puede utilizar para determinar los tipos de datos a los que se puede promover un tipo de datos determinado. La tabla muestra que la mejor elección siempre es el mismo tipo de datos en lugar de elegir la promoción a otro tipo de datos.

Tabla 5. Tabla de prioridades de tipos de datos

Tipo de datos	Lista de prioridad de tipos de datos (por orden de mejor a peor)
CHAR	CHAR, VARCHAR, LONG VARCHAR, CLOB
VARCHAR	VARCHAR, LONG VARCHAR, CLOB
LONG VARCHAR	LONG VARCHAR, CLOB
GRAPHIC	GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, DBCLOB
VARGRAPHIC	VARGRAPHIC, LONG VARGRAPHIC, DBCLOB
LONG VARGRAPHIC	LONG VARGRAPHIC, DBCLOB
BLOB	BLOB
CLOB	CLOB
DBCLOB	DBCLOB
SMALLINT	SMALLINT, INTEGER, BIGINT, decimal, real, double
INTEGER	INTEGER, BIGINT, decimal, real, double
BIGINT	BIGINT, decimal, real, double

Tabla 5. Tabla de prioridades de tipos de datos (continuación)

Tipo de datos	Lista de prioridad de tipos de datos (por orden de mejor a peor)
decimal	decimal, real, double
real	real, double
double	double
DATE	DATE
TIME	TIME
TIMESTAMP	TIMESTAMP
DATALINK	DATALINK
udt	udt (mismo nombre) o un supertipo de udt
REF(T)	REF(S) (en el caso de que S sea un supertipo de T)

Nota:

Los tipos en minúsculas que aparecen en la lista se definen de la siguiente manera:

decimal

= DECIMAL(p,s) o NUMERIC(p,s)

real

= REAL o FLOAT(n) donde n no es mayor que 24

double

= DOUBLE, DOUBLE-PRECISION, FLOAT o FLOAT(n) donde n es mayor que 24

udt

= un tipo definido por el usuario

Los sinónimos, más cortos o más largos, de los tipos de datos listados se consideran iguales al sinónimo listado.

Conversión entre tipos de datos

En muchas ocasiones, un valor con un tipo de datos determinado necesita *convertirse* a un tipo de datos diferente o al mismo tipo de datos con una longitud, precisión o escala diferentes. La promoción del tipo de datos (tal como se define en “Promoción de los tipos de datos” en la página 100) es un ejemplo en el que la promoción de un tipo de datos a otro tipo de datos necesita que el valor se convierta al nuevo tipo de datos. Un tipo de datos que se puede convertir a otro tipo de datos es *convertible* de un tipo de datos fuente al tipo de datos de destino.

La conversión entre tipos de datos puede realizarse explícitamente utilizando la especificación CAST (vea “Especificaciones CAST” en la página 193) pero también puede efectuarse implícitamente durante las asignaciones que implican tipos definidos por el usuario (vea “Asignaciones de los tipos definidos por el usuario” en la página 113). También, cuando se crean funciones definidas por el usuario derivadas (vea “CREATE FUNCTION” en la página 695

Conversión entre tipos de datos

la página 695), los tipos de datos de los parámetros de la función fuente deben poder convertirse a los tipos de datos de la función que se está creando.

La Tabla 6 en la página 104 muestra las conversiones permitidas entre tipos de datos internos.

Se da soporte a las siguientes conversiones en las que intervienen tipos diferenciados:

- conversión de un tipo diferenciado *DT* en su tipo de datos fuente *S*
- conversión del tipo de datos fuente *S* de un tipo diferenciado *DT* en el tipo diferenciado *DT*
- conversión del tipo diferenciado *DT* en el mismo tipo diferenciado *DT*
- conversión de un tipo de datos *A* en un tipo diferenciado *DT* donde *A* se puede promocionar al tipo de datos fuente *S* del tipo diferenciado *DT* (vea “Promoción de los tipos de datos” en la página 100)
- conversión de INTEGER en un tipo diferenciado *DT* con un tipo de datos fuente SMALLINT
- conversión de DOUBLE en un tipo diferenciado *DT* con un tipo de datos fuente REAL
- conversión de VARCHAR en un tipo diferenciado *DT* con un tipo de datos fuente CHAR
- conversión de VARGRAPHIC en un tipo diferenciado *DT* con un tipo de datos fuente GRAPHIC.

No es posible especificar FOR BIT DATA cuando se realiza una conversión a un tipo de carácter.

No es posible convertir un valor de tipo estructurado en algo diferente. Un tipo estructurado *ST* no necesita convertirse a uno de sus supertipos, pues todos los métodos de los supertipos de *ST* son aplicables a *ST*. Si la operación deseada sólo es aplicable a un subtipo de *ST*, entonces utilice la expresión de tratamiento de subtipos para tratar *ST* como uno de sus subtipos. Vea “Tratamiento de los subtipos” en la página 206 para obtener detalles.

Cuando un tipo de datos definido por el usuario e implicado en una conversión no está calificado por un nombre de esquema, se utiliza la *vía de acceso de SQL* para buscar el primer esquema que incluya el tipo de datos definido por el usuario con este nombre. La vía de acceso de SQL se describe con más detalle en “CURRENT PATH” en la página 138.

Se da soporte a las siguientes conversiones donde intervienen tipos de referencia:

- conversión de un tipo de referencia *RT* en su tipo de datos de representación *S*

Conversión entre tipos de datos

- conversión del tipo de datos de representación S de un tipo de referencia RT en el tipo de referencia RT
- conversión de un tipo de referencia RT con un tipo de destino T en un tipo de referencia RS con un tipo de destino S donde S es un supertipo de T .
- conversión de un tipo de datos A en un tipo de referencia RT donde A se puede promocionar al tipo de datos de representación S del tipo de referencia RT (consulte “Promoción de los tipos de datos” en la página 100).

Cuando el tipo de destino de un tipo de datos de referencia implicado en una conversión no está calificado por un nombre de esquema, se utiliza la *vía de acceso de SQL* para buscar el primer esquema que incluya el tipo de datos definido por el usuario con este nombre. La vía de acceso de SQL se describe con más detalle en “CURRENT PATH” en la página 138.

Conversión entre tipos de datos

Tabla 6. Conversiones soportadas entre tipos de datos internos

Tipo de datos de destino →	S	I	B	D	R	D	C	V	L	C	G	V	L	D	D	T	T	B
Tipo de datos fuente ↓	M	N	I	E	E	O	H	A	O	L	R	A	O	B	A	I	I	L
	A	T	G	C	A	U	A	R	N	O	A	R	N	C	T	M	M	O
	L	E	I	I	L	B	R	C	G	B	P	G	G	L	E	E	E	B
	L	G	N	M	L	H	V	H	R	V	H	R	V	O			S	
	I	E	T	A	E	A	A	I	A	A	I	A	A	B			T	
	N	R		L			R	R	C	P	R						A	
	T							C	H	G							M	
								H	I								P	
								A										
								R										
SMALLINT	Y	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-
INTEGER	Y	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-
BIGINT	Y	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-
DECIMAL	Y	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-
REAL	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-	-
DOUBLE	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-	-
CHAR	Y	Y	Y	Y	-	-	Y	Y	Y	Y	-	Y	-	-	Y	Y	Y	Y
VARCHAR	Y	Y	Y	Y	-	-	Y	Y	Y	Y	-	Y	-	-	Y	Y	Y	Y
LONG VARCHAR	-	-	-	-	-	-	Y	Y	Y	Y	-	-	-	-	-	-	-	Y
CLOB	-	-	-	-	-	-	Y	Y	Y	Y	-	-	-	-	-	-	-	Y
GRAPHIC	-	-	-	-	-	-	-	-	-	-	Y	Y	Y	Y	-	-	-	Y
VARGRAPHIC	-	-	-	-	-	-	-	-	-	-	Y	Y	Y	Y	-	-	-	Y
LONG VARG	-	-	-	-	-	-	-	-	-	-	Y	Y	Y	Y	-	-	-	Y
DBCLOB	-	-	-	-	-	-	-	-	-	-	Y	Y	Y	Y	-	-	-	Y
DATE	-	-	-	-	-	-	Y	Y	-	-	-	-	-	-	Y	-	-	-
TIME	-	-	-	-	-	-	Y	Y	-	-	-	-	-	-	-	Y	-	-
TIMESTAMP	-	-	-	-	-	-	Y	Y	-	-	-	-	-	-	Y	Y	Y	-
BLOB	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	Y

Notas

- Vea la descripción que precede a la tabla para conocer las conversiones soportadas donde intervienen tipos definidos por el usuario y tipos de referencia.
- Sólo un tipo DATALINK puede convertirse a un tipo DATALINK.
- No es posible convertir un valor de tipo estructurado en algo diferente.

Asignaciones y comparaciones

Las operaciones básicas de SQL son la asignación y la comparación. Las operaciones de asignación se llevan a cabo durante la ejecución de las sentencias de variables de transición INSERT, UPDATE, FETCH, SELECT INTO, VALUES INTO y SET. Los argumentos de las funciones también se asignan cuando se invoca una función. Las operaciones de comparación se realizan durante la ejecución de las sentencias que incluyen predicados y otros elementos del lenguaje como, por ejemplo, MAX, MIN, DISTINCT, GROUP BY y ORDER BY.

La regla básica para las dos operaciones es que el tipo de datos de los operandos implicados debe ser compatible. La regla de compatibilidad también se aplica a las operaciones de conjuntos (vea “Reglas para los tipos de datos del resultado” en la página 120). La matriz de compatibilidad es la siguiente.

Tabla 7. Compatibilidad de tipos de datos para asignaciones y comparaciones

Operandos	Entero binario	Número decimal	Coma flotante	Serie de caracteres	Serie gráfica	Fecha	Hora	Indicación de la hora	Serie binaria	UDT
Entero binario	Sí	Sí	Sí	No	No	No	No	No	No	²
Número decimal	Sí	Sí	Sí	No	No	No	No	No	No	²
Coma flotante	Sí	Sí	Sí	No	No	No	No	No	No	²
Serie de caracteres	No	No	No	Sí	No	¹	¹	¹	No ³	²
Serie gráfica	No	No	No	No	Sí	No	No	No	No	²
Fecha	No	No	No	¹	No	Sí	No	No	No	²
Hora	No	No	No	¹	No	No	Sí	No	No	²
Indicación de la hora	No	No	No	¹	No	No	No	Sí	No	²
Serie binaria	No	No	No	No ³	No	No	No	No	Sí	²
UDT	²	²	²	²	²	²	²	²	²	Sí

Asignaciones y comparaciones

Tabla 7. Compatibilidad de tipos de datos para asignaciones y comparaciones (continuación)

Operandos	Entero binario	Número decimal	Coma flotante	Serie de caracteres	Serie gráfica	Fecha	Hora	Indicación de la hora	Serie binaria	UDT
-----------	----------------	----------------	---------------	---------------------	---------------	-------	------	-----------------------	---------------	-----

Nota:

- ¹ La compatibilidad de los valores de indicación de fecha y hora y series de caracteres está limitada a la asignación y la comparación:
 - Los valores de indicación de fecha y hora se pueden asignar a las columnas de series de caracteres y a las variables de series de caracteres tal como se explica en el apartado “Asignaciones de indicación de fecha y hora” en la página 110.
 - Una representación de serie válida de una fecha se puede asignar a una columna de fecha o comparar con una fecha.
 - Una representación de serie válida de una hora se puede asignar a una columna de hora o comparar con una hora.
 - Una representación de serie válida de una indicación de la hora se puede asignar a una columna de indicación de la hora o comparar con una indicación de la hora.
- ² Un valor de tipo diferenciado definido por el usuario (valor UDDT) sólo se puede comparar con un valor definido con el mismo UDDT. En general, se da soporte a las asignaciones entre un valor de tipo diferenciado y su tipo de datos fuente. Un tipo estructurado definido por el usuario no es comparable y sólo se puede asignar a un operando del mismo tipo estructurado o a uno de sus subtipos. Para obtener más información, vea “Asignaciones de los tipos definidos por el usuario” en la página 113.
- ³ Observe que esto significa que las series de caracteres definidas con el atributo FOR BIT DATA tampoco son compatibles con las series binarias.
- ⁴ Un operando DATALINK sólo puede asignarse a otro operando DATALINK. El valor DATALINK sólo puede asignarse a una columna si la columna está definida con NO LINK CONTROL o el archivo existe y todavía no está bajo el control del enlace del archivo.
- ⁵ Para obtener información sobre la asignación y comparación de tipos de referencia, consulte “Asignaciones de tipos de referencia” en la página 114 y “Comparaciones de tipos de referencia” en la página 120.

Una regla básica para la operaciones de asignación es que un valor nulo no puede asignarse a una columna que no pueda contener valores nulos, ni a una variable del lenguaje principal que no tenga una variable indicadora asociada. (Consulte el apartado “Referencias a variables del lenguaje principal” en la página 151 para ver una explicación de las variables indicadoras.)

Asignaciones numéricas

La regla básica para las asignaciones numéricas es que la parte entera de un número decimal o entero no se trunca nunca. Si la escala del número de destino es menor que la escala del número asignado, se trunca el exceso de dígitos de la fracción de un número decimal.

De decimal o entero a coma flotante

Los números de coma flotante son aproximaciones de números reales. Por lo tanto, cuando se asigna un número decimal o entero a una columna variable o de coma flotante, es posible que el resultado no sea idéntico al número original.

De coma flotante o decimal a entero

Cuando un número de coma flotante o decimal se asigna a una columna o variable de enteros, se pierde la parte correspondiente a la fracción del número.

De decimal a decimal

Cuando se asigna un número decimal a una columna o variable decimal, el número se convierte, si es necesario, a la precisión y la escala del destino. Se añade o elimina el número necesario de ceros iniciales y, en la fracción del número, se añaden los ceros de cola necesarios o se eliminan los dígitos de cola necesarios.

De entero a decimal

Cuando se asigna un entero a una columna o variable decimal, primero el número se convierte a un número decimal temporal y, después, si es necesario, a la precisión y escala del destino. La precisión y escala del número decimal temporal es de 5,0 para un entero pequeño, 11,0 para un entero grande o 19,0 para un entero superior.

De coma flotante a decimal

Cuando se convierte un número de coma flotante a decimal, primero el número se convierte a un número decimal temporal de precisión 31 y, después si es necesario, se trunca a la precisión y escala del destino. En esta conversión, el número se redondea (utilizando la aritmética de coma flotante) a una precisión de 31 dígitos decimales. Como resultado, los números menores que $0,5 \cdot 10^{-31}$ se reducen a 0. Se da el valor más grande posible a la escala que permita representar la parte entera del número sin pérdida de significación.

Asignaciones de series

Existen dos tipos de asignaciones:

- la *asignación de almacenamiento* es cuando se asigna un valor a una columna o parámetro de una función
- la *asignación de recuperación* es cuando se asigna un valor a una variable del lenguaje principal.

Las reglas para la asignación de series difieren según el tipo de asignación.

Asignación de almacenamiento

La regla básica es que la longitud de la serie asignada a una columna o a un parámetro de función no debe ser mayor que el atributo de longitud de la

Asignaciones y comparaciones

columna o del parámetro de función. Cuando la longitud de la serie es mayor que el atributo de longitud de la columna o del parámetro de función, se pueden producir las siguientes situaciones:

- la serie se asigna con los blancos de cola truncados (de todos los tipos de serie excepto de las series largas) para ajustarse al atributo de longitud de la columna o del atributo de función de destino
- se devuelve un error (SQLSTATE 22001) cuando:
 - se truncarían caracteres que no son blancos de una serie que no es larga
 - se truncaría cualquier carácter (o byte) de una serie larga.

Cuando se asigna una serie a una columna de longitud fija y la longitud de la serie es menor que el atributo de longitud del destino, la serie se rellena por la derecha con el número necesario de blancos. El carácter de relleno siempre es el blanco incluso para las columnas definidas con el atributo FOR BIT DATA.

Asignación de recuperación

La longitud de una serie asignada a una variable del lenguaje principal puede ser mayor que el atributo de longitud de la variable del lenguaje principal. Cuando una serie se asigna a una variable del lenguaje principal y la longitud de la serie es mayor que la longitud del atributo de longitud de la variable, la serie se trunca por la derecha el número necesario de caracteres (o bytes). Cuando ocurre esto, se devuelve un aviso (SQLSTATE 01004) y se asigna el valor 'W' al campo SQLWARN1 de la SQLCA.

Además, si se proporciona una variable indicadora y la fuente del valor no es LOB, la variable indicadora se establece en la longitud original de la serie.

Cuando se asigna una serie de caracteres a una variable de longitud fija y la longitud de la serie es menor que el atributo de longitud del destino, la serie se rellena por la derecha con el número necesario de blancos. El carácter de relleno siempre es un blanco incluso para las series definidas con el atributo FOR BIT DATA.

La asignación de recuperación de las variables del lenguaje principal terminadas en nulo en C se maneja en base a las opciones especificadas con los mandatos PREP o BIND. Consulte la sección sobre programación en C y en C++ en el manual *Application Development Guide* para obtener detalles.

Reglas de conversión para asignaciones de series

Una serie de caracteres o una serie gráfica asignada a una columna o a una variable del lenguaje principal se convierte primero, si es necesario, a la página de códigos del destino. La conversión de caracteres sólo es necesaria si son ciertas todas las condiciones siguientes:

- Las páginas de códigos son diferentes.

- La serie no es nula ni está vacía.
- Ninguna serie tiene un valor de página de códigos de 0 (FOR BIT DATA).¹¹

Consideraciones acerca de MBCS para las asignaciones de series de caracteres

Existen varias consideraciones al asignar series de caracteres que pueden contener caracteres de un solo byte y de múltiples bytes. Estas consideraciones se aplican a todas las series de caracteres, incluyendo las definidas como FOR BIT DATA.

- El relleno con blancos siempre se realiza utilizando el carácter blanco de un solo byte (X'20').
- El truncamiento de blancos siempre se realiza en base al carácter blanco de un solo byte (X'20'). El carácter blanco de doble byte se trata como cualquier otro carácter con respecto al truncamiento.
- La asignación de una serie de caracteres a una variable del lenguaje principal puede dar como resultado la fragmentación de caracteres MBCS si la variable del lenguaje principal de destino no es lo suficientemente grande para contener toda la serie fuente. Si se fragmenta un carácter MBCS, cada byte del fragmento del carácter MBCS destino se establece en el destino en un carácter blanco de un solo byte (X'20'), no se mueven más bytes de la fuente y SQLWARN1 se establece en 'W' para indicar el truncamiento. Observe que se aplica el mismo manejo de fragmentos de caracteres MBCS incluso cuando la serie de caracteres está definida como FOR BIT DATA.

Consideraciones acerca de DBCS para las asignaciones de series gráficas

Las asignaciones de series gráficas se procesan de manera análoga a la de las series de caracteres. Los tipos de datos de serie gráfica sólo son compatibles con otros tipos de datos de series gráficas y nunca con tipos de datos numéricos, de serie de caracteres o de indicación de fecha y hora.

Si se asigna un valor de serie gráfica a una columna de serie gráfica, la longitud del valor no debe ser mayor que la longitud de la columna.

Si un valor de serie gráfica (la serie 'fuente') se asigna a un tipo de datos de serie gráfica de longitud fija (el 'destino', que puede ser una columna o una variable del lenguaje principal), y la longitud de la serie fuente es menor que el destino, éste contendrá una copia de la serie fuente que se habrá rellenado por la derecha con el número necesario de caracteres blancos de doble byte para crear un valor cuya longitud sea igual a la del destino.

11. Cuando actúa como un servidor de aplicaciones DRDA, las variables del lenguaje principal se convierten a la página de códigos del servidor de aplicaciones, incluso si se están asignando, comparando o combinando con una columna FOR BIT DATA. Si la SQLDA se ha modificado para identificar la variable del lenguaje principal de entrada como FOR BIT DATA, no se lleva a cabo la conversión.

Asignaciones y comparaciones

Si se asigna un valor de serie gráfica a una variable de lenguaje principal de serie gráfica y la longitud de la serie fuente es mayor que la longitud de la variable del lenguaje principal, ésta contendrá una copia de la serie fuente que se habrá truncado por la derecha el número necesario de caracteres de doble byte para crear un valor cuya longitud sea igual al de la variable del lenguaje principal. (Tenga en cuenta que para este caso, es necesario que el truncamiento no implique la bisección de un carácter de doble byte; si hubiera que realizar una bisección, el valor fuente o la variable del lenguaje principal de destino tendrían un tipo de datos de serie gráfica mal definido.) El distintivo de aviso SQLWARN1 del SQLCA se establecerá en 'W'. La variable indicadora, si se especifica, contendrá la longitud original (en caracteres de doble byte) de la serie fuente. Sin embargo, en el caso de DBCLOB, la variable indicadora no contiene la longitud original.

La asignación de recuperación de las variables del lenguaje principal terminadas en nulo en C (declaradas utilizando wchar_t) se maneja sobre la base de las opciones especificadas con los mandatos PREP o BIND. Consulte la sección sobre programación en C y en C++ del manual *Application Development Guide* para obtener detalles.

Asignaciones de indicación de fecha y hora

La regla básica para las asignaciones de indicación de fecha y hora es que un valor DATE, TIME o TIMESTAMP sólo puede asignarse a una columna con un tipo de datos coincidente (ya sea DATE, TIME o TIMESTAMP) o a una variable de serie de caracteres o columna de serie de longitud variable o fija. La asignación no debe ser a una variable o columna LONG VARCHAR, BLOB ni CLOB.

Cuando un valor de indicación de fecha y hora se asigna a una variable de serie de caracteres o a una columna de serie, la conversión a una representación de serie es automática. Los ceros iniciales no se omiten de ninguna parte de la fecha, de la hora ni de la indicación de la hora. La longitud necesaria del destino variará, según el formato de la representación de serie. Si la longitud del destino es mayor que la necesaria y el destino es una serie de longitud fija, se rellena por la derecha con blancos. Si la longitud del destino es menor que la necesaria, el resultado depende del tipo del valor de indicación de fecha y hora implicado y del tipo de destino.

Cuando el destino es una variable del lenguaje principal, se aplican las reglas siguientes:

- **Para DATE:** Si la longitud de la variable es menor que 10 bytes, se produce un error.
- **Para TIME:** Si se utiliza el formato USA, la longitud de la variable no debe ser menor de 8; en otros formatos la longitud no debe ser menor que 5.

Si se utilizan los formatos ISO o JIS, y la longitud de la variable del lenguaje principal es menor que 8, la parte correspondiente a los segundos de la hora se omite del resultado y se asigna a la variable indicadora, si se proporciona. El campo SQLWARN1 de la SQLCA se establece de manera que indica la omisión.

- **Para TIMESTAMP:** Si la variable del lenguaje principal es menor que 19 bytes, se produce un error. Si la longitud es menor que 26, pero mayor o igual que 19 bytes, los dígitos de cola de la parte correspondiente a los microsegundos del valor se omiten. El campo SQLWARN1 de la SQLCA se establece de manera que indica la omisión.

Para obtener más información acerca de las longitudes de serie para los valores de indicación de fecha y hora, vea “Valores de indicación de fecha y hora” en la página 91.

Asignaciones de DATALINK

La asignación de un valor a una columna DATALINK da como resultado el establecimiento de un enlace con un archivo a no ser que los atributos de enlace del valor estén vacíos o que la columna esté definida con NO LINK CONTROL. En los casos en que ya exista un valor enlazado en la columna, este archivo queda desenlazado. La asignación de un valor nulo donde ya existe un valor enlazado también desenlaza el archivo asociado con el valor anterior.

Si la aplicación proporciona la misma ubicación de datos que la que ya existe en la columna, se retiene el enlace. Hay dos razones para que pueda producirse este hecho:

- se cambia el comentario
- si la tabla se coloca en el estado de reconciliación de Datalink no posible (DRNP), pueden restablecerse los enlaces en la tabla al proporcionarse unos atributos de enlace idénticos a los de la columna.

Un valor DATALINK puede asignarse a una columna de cualquiera de las maneras siguientes:

- Se puede utilizar la función escalar DLVALUE para crear un nuevo valor DATALINK y asignarlo a una columna. A no ser que el valor sólo contenga un comentario o que el URL sea exactamente el mismo, la acción de asignación enlazará el archivo.
- Se puede crear un valor DATALINK en un parámetro de CLI con la función SQLBuildDataLink de CLI. A continuación, este valor puede asignarse a una columna. A no ser que el valor sólo contenga un comentario o que el URL sea exactamente el mismo, la acción de asignación enlazará el archivo.

Cuando se asigna un valor a una columna DATALINK, las siguientes condiciones de error devuelven SQLSTATE 428D1:

Asignaciones y comparaciones

- El formato de la Ubicación de datos (URL) no es válido (código de razón 21).
- El servidor de archivos no está registrado con esta base de datos (código de razón 22).
- El tipo de enlace especificado no es válido (código de razón 23).
- La longitud del comentario o del URL no es válida (código de razón 27).

Tenga en cuenta que el tamaño de un resultado de función o parámetro de URL es el mismo en la entrada y en la salida y que dicho tamaño está limitado por la longitud de la columna DATALINK. Sin embargo, en algunos casos el valor del URL se devuelve con un símbolo de acceso adjunto. En las situaciones en que esto sea posible, la ubicación de la salida debe tener espacio de almacenamiento suficiente para el símbolo de accesos y la longitud de la columna DATALINK. En consecuencia, la longitud real del comentario y del URL (en su formato totalmente ampliado) que se proporciona en la entrada debe restringirse para adaptarse al espacio de almacenamiento de la salida. Si se sobrepasa la longitud restringida, surge este error.

Cuando la asignación también vaya a crear un enlace, pueden producirse los errores siguientes:

- El servidor de archivos no está disponible actualmente (SQLSTATE 57050).
- El archivo no existe (SQLSTATE 428D1, código de razón 24).
- No se puede acceder al archivo de referencia para el enlace (código de razón 26).
- El archivo ya está enlazado con otra columna (SQLSTATE 428D1, código de razón 25).

Tenga en cuenta que surgirá este error aunque el enlace sea con una base de datos diferente.

Además, cuando la asignación elimine un enlace existente, pueden producirse los errores siguientes:

- El servidor de archivos no está disponible actualmente (SQLSTATE 57050).
- El archivo con control de integridad referencial no presenta un estado correcto según Data Links File Manager (SQLSTATE 58004).

Puede recuperarse un valor DATALINK de la base de datos de cualquiera de las maneras siguientes:

- Pueden asignarse las partes de un valor DATALINK a variables del lenguaje principal con la utilización de funciones escalares (como DLLINKTYPE o DLURLPATH).

Tenga en cuenta que, normalmente, no se intenta acceder al servidor de archivos en el momento de la recuperación.¹² Por lo tanto, es posible que fallen los intentos subsiguientes de acceso al servidor de archivos mediante mandatos de sistema de archivos.

Cuando se recupera un valor DATALINK, se comprueba el registro de servidores de archivos en el servidor de bases de datos para confirmar que el servidor de archivos todavía está registrado con el servidor de bases de datos (SQLSTATE 55022). Además, es posible que se devuelva un aviso cuando se recupere un valor DATALINK por encontrarse la tabla en un estado pendiente de reconciliación o en un estado de reconciliación no posible (SQLSTATE 01627).

Asignaciones de los tipos definidos por el usuario

Con los tipos definidos por el usuario, se aplican diferentes reglas para las asignaciones a variables del lenguaje principal que se utilizan para todas las asignaciones.

Tipos diferenciados: La asignación a variables del lenguaje principal se realiza basándose en el tipo fuente del tipo diferenciado. Es decir, sigue la regla:

- Un valor de un tipo diferenciado en el lado derecho de una asignación se puede asignar a una variable del lenguaje principal en el lado izquierdo sólo si el tipo fuente del tipo diferenciado se puede asignar a la variable del lenguaje principal.

Si el destino de la asignación es una columna basada en un tipo diferenciado, el tipo de datos fuente debe ser convertible al tipo de datos de destino, tal como se describe en “Conversión entre tipos de datos” en la página 101 para los tipos definidos por el usuario.

Tipos estructurados: La asignación realizada con variables del lenguaje principal está basada en el tipo declarado de la variable del lenguaje principal. Es decir, sigue esta regla:

Un valor de un tipo estructurado situado en el lado derecho de una asignación se puede asignar a una variable de lenguaje principal, situada en el lado izquierdo, sólo si el tipo declarado de la variable es el tipo estructurado o un supertipo del tipo estructurado.

12. Puede que sea necesario acceder al servidor de archivos para determinar el nombre de prefijo asociado con una vía de acceso. Se puede cambiar en el servidor de archivos cuando se mueva el punto de montaje de un sistema de archivos. El primer acceso a un archivo de un servidor causará que se recuperen los valores necesarios del servidor de archivos y se coloquen en la antememoria del servidor de bases de datos para la subsiguiente recuperación de valores DATALINK de este servidor de archivos. Se devuelve un error si no se puede acceder al servidor de archivos (SQLSTATE 57050).

Asignaciones y comparaciones

Si el destino de la asignación es una columna de un tipo estructurado, el tipo de datos fuente debe ser el tipo de datos destino o un subtipo de él.

Asignaciones de tipos de referencia

Un tipo de referencia cuyo tipo destino sea T se puede asignar a una columna de tipo de referencia que también es un tipo de referencia cuyo tipo destino sea S , donde S es un supertipo de T . Si se realiza una asignación a una columna o variable de referencia con ámbito, no tiene lugar ninguna comprobación para asegurarse de que el valor real que se asigna exista en la tabla o vista de destino definidos por el ámbito.

La asignación a variables del lenguaje principal tiene lugar sobre la base del tipo de representación del tipo de referencia. Es decir, sigue la regla:

- Un valor de un tipo de referencia del lado derecho de una asignación puede asignarse a una variable del lenguaje principal del lado izquierdo si y sólo si el tipo de representación de este tipo de referencia puede asignarse a esta variable del lenguaje principal.

Si el destino de la asignación es una columna y el lado derecho de la asignación es una variable del lenguaje principal, la variable del lenguaje principal debe convertirse explícitamente al tipo de referencia de la columna de destino.

Comparaciones numéricas

Los números se comparan algebraicamente; es decir, tomando en consideración el signo. Por ejemplo, -2 es menor que $+1$.

Si un número es un entero y el otro es un decimal, la comparación se realiza con una copia temporal del entero, que se ha convertido a decimal.

Cuando se comparan números decimales con diferentes escalas, la comparación se realiza con una copia temporal de uno de los números que se ha extendido con ceros de cola para que su parte correspondiente a la fracción tenga el mismo número de dígitos que el otro número.

Si un número es de coma flotante y el otro es un entero o un decimal, la comparación se efectúa con una copia temporal del otro número, que se ha convertido a coma flotante de doble precisión.

Dos números de coma flotante sólo son iguales si las configuraciones de bits de sus formatos normalizados son idénticos.

Comparaciones de series

Las series de caracteres se comparan de acuerdo con el orden de clasificación especificado cuando se ha creado la base de datos, excepto aquellos con el atributo FOR BIT DATA que siempre se comparan de acuerdo con sus valores de bits.

Cuando se comparan series de caracteres de longitudes desiguales, la comparación se realiza utilizando una copia lógica de la serie más corta que se ha rellenado por la derecha con suficientes blancos de un solo byte para extender su longitud a la de la serie más larga. Esta extensión lógica se realiza para todas las series de caracteres incluyendo las que tienen el distintivo FOR BIT DATA.

Las series de caracteres (excepto las series de caracteres con el distintivo FOR BIT DATA) se comparan de acuerdo con el orden de clasificación especificado cuando se ha creado la base de datos (consulte el manual *Administration Guide* para obtener más información acerca de los órdenes de clasificación especificados en el momento de la creación de la base de datos). Por ejemplo, el orden de clasificación por omisión suministrado por el gestor de bases de datos puede dar el mismo peso a versiones en minúsculas y en mayúsculas del mismo carácter. El gestor de bases de datos realiza una comparación en dos pasos para asegurarse de que sólo se consideran iguales las series idénticas. En el primer paso, las series se comparan de acuerdo con el orden de clasificación de la base de datos. Si los pesos de los caracteres de las series son iguales, se realiza un segundo paso de "desempate" para comparar las series en base a sus valores de elemento de código real.

Dos series son iguales si las dos están vacías o si todos los bytes correspondientes son iguales. Si cualquier operando es nulo, el resultado es desconocido.

No se da soporte a las series largas ni a las series LOB las operaciones de comparación que utilizan operadores de comparación básicos (=, <>, <, >, <= y >=). Están soportadas en comparaciones que utilizan el predicado LIKE y la función POSSTR. Consulte los detalles en el apartado "Predicado LIKE" en la página 223 y en el apartado "POSSTR" en la página 395.

Los fragmentos de series largas y series LOB de un máximo de 4.000 bytes se pueden comparar utilizando las funciones escalares SUBSTR y VARCHAR. Por ejemplo, si se tienen las columnas:

```
MI_SHORT_CLOB    CLOB(300)
MI_LONG_VAR      LONG VARCHAR
```

entonces lo siguiente será válido:

```
WHERE VARCHAR(MI_SHORT_CLOB) > VARCHAR(SUBSTR(MI_LONG_VAR,1,300))
```

Asignaciones y comparaciones

Ejemplos:

Para estos ejemplos, 'A', 'Á', 'a' y 'á', tienen los valores de elemento de código X'41', X'C1', X'61' y X'E1' respectivamente.

Considere un orden de clasificación en el que los caracteres 'A', 'Á', 'a', 'á' tengan los pesos 136, 139, 135 y 138. Entonces, los caracteres se clasifican en el orden de sus pesos de la forma siguiente:

'a' < 'A' < 'á' < 'Á'

Ahora considere cuatro caracteres DBCS D1, D2, D3 y D4 con los elementos de código 0xC141, 0xC161, 0xE141 y 0xE161, respectivamente. Si estos caracteres DBCS están en columnas CHAR, se clasifican como una secuencia de bytes según los pesos de clasificación de estos bytes. Los primeros bytes tienen pesos de 138 y 139, por consiguiente D3 y D4 vienen antes que D2 y D1; los segundos bytes tienen pesos de 135 y 136. Por consiguiente, el orden es el siguiente:

D4 < D3 < D2 < D1

Sin embargo, si los valores que se comparan tienen el atributo FOR BIT DATA o si estos caracteres DBCS se guardaron en una columna GRAPHIC, los pesos de clasificación no se tienen en cuenta y los caracteres se comparan de acuerdo con los elementos de código del modo siguiente:

'A' < 'a' < 'Á' < 'á'

Los caracteres DBCS se clasifican como secuencia de bytes, en el orden de los elementos de código del modo siguiente:

D1 < D2 < D3 < D4

Ahora considere un orden de clasificación en el que los caracteres 'A', 'Á', 'a', 'á' tengan los pesos (no exclusivos) 74, 75, 74 y 75. Si consideramos únicamente los pesos de clasificación (primer pase), 'a' es igual a 'A' y 'á' es igual a 'Á'. Los elementos de código de los caracteres se utilizan para romper el empate (segundo pase) del modo siguiente:

'A' < 'a' < 'Á' < 'á'

Los caracteres DBCS de las columnas CHAR se clasifican como una secuencia de bytes, de acuerdo con sus pesos (primer pase) y luego de acuerdo con los elementos de código para romper el empate (segundo pase). Los primeros bytes tienen pesos iguales, por lo tanto los elementos de código (0xC1 y 0xE1) rompen el empate. Por consiguiente, los caracteres D1 y D2 se clasifican antes que los caracteres D3 y D4. A continuación, se comparan los segundos bytes de una forma similar y el resultado es el siguiente:

D1 < D2 < D3 < D4

Una vez más, si los datos de las columnas CHAR tienen el atributo FOR BIT DATA o si los caracteres DBCS se guardan en una columna GRAPHIC, los pesos de clasificación no se tienen en cuenta y se comparan los caracteres de acuerdo con los elementos de código:

D1 < D2 < D3 < D4

Para este ejemplo en concreto, el resultado parece ser el mismo que cuando se utilizaron los pesos de clasificación, pero obviamente, éste no siempre es el caso.

Reglas de conversión para la comparación

Cuando se comparan dos series, primero una de ellas se convierte, si es necesario, al juego de la página de códigos de la otra serie. Para obtener detalles, vea “Reglas para las conversiones de series” en la página 124.

Clasificación de los resultados

Los resultados que necesitan clasificarse se ordenan en base a las reglas de comparación de series explicadas en el apartado “Comparaciones de series” en la página 115. La comparación se efectúa en el servidor de bases de datos. Al devolver los resultados a la aplicación cliente, se puede llevar a cabo una conversión de la página de códigos. Esta conversión de la página de códigos subsiguiente no afecta al orden del conjunto resultante determinado por el servidor.

Consideraciones acerca de MBCS para las comparaciones de series

Las series de caracteres SBCS/MBCS mixtas se comparan de acuerdo con el orden de clasificación especificado cuando se ha creado la base de datos. Para las bases de datos creadas con el orden de clasificación (SYSTEM) por omisión, todos los caracteres ASCII de un solo byte se clasifican según el orden correcto, pero los caracteres de doble byte no se encuentran necesariamente por orden de elemento de código. Para las bases de datos creadas con el orden IDENTITY, todos los caracteres de doble byte se clasifican correctamente por orden de elemento de código, pero los caracteres ASCII de un solo byte también se clasifican por orden de elemento de código. Para las bases de datos creadas con el orden COMPATIBILITY, se utiliza un orden de acomodación que clasifica correctamente la mayoría de los caracteres de doble byte y resulta casi correcto para ASCII. Ésta era la tabla de clasificación por omisión en DB2 Versión 2.

Las series de caracteres mixtas se comparan byte a byte. Esto puede provocar resultados anómalos para los caracteres de múltiples bytes que aparezcan en series mixtas, porque cada byte se toma en cuenta independientemente.

Ejemplo:

Asignaciones y comparaciones

Para este ejemplo, los caracteres de doble byte 'A', 'B', 'a' y 'b' tienen los valores de elemento de código X'8260', X'8261', X'8281' y X'8282', respectivamente.

Considere un orden de clasificación en el que los elementos de código X'8260', X'8261', X'8281' y X'8282' tengan los pesos 96, 65, 193 y 194. Entonces:

'B' < 'A' < 'a' < 'b'

y

'AB' < 'AA' < 'Aa' < 'Ab' < 'aB' < 'aA' < 'aa' < 'ab'

Las comparaciones de series gráficas se procesan de manera análoga a la de las series de caracteres.

Las comparaciones de series gráficas son válidas entre todos los tipos de datos de series gráficas excepto LONG VARCHAR. Los tipos de datos LONG VARCHAR y DBCLOB no están permitidos en una operación de comparación.

Para series gráficas, el orden de clasificación de la base de datos no se utiliza. En su lugar, las series gráficas se comparan siempre en base a los valores numéricos (binarios) de sus bytes correspondientes.

Utilizando el ejemplo anterior, si los literales fuesen series gráficas, entonces:

'A' < 'B' < 'a' < 'b'

y

'AA' < 'AB' < 'Aa' < 'Ab' < 'aA' < 'aB' < 'aa' < 'ab'

Cuando se comparan series gráficas de longitudes distintas, la comparación se realiza utilizando una copia lógica de la serie más corta que se ha rellenado por la derecha con suficientes caracteres blancos de doble byte para extender su longitud a la de la serie más larga.

Dos valores gráficos son iguales si los dos están vacíos o si todos los gráficos correspondientes son iguales. Si cualquier operando es nulo, el resultado es desconocido. Si dos valores no son iguales, su relación se determina por una simple comparación de series binarias.

Tal como se indica en esta sección, la comparación de series byte a byte puede producir resultados insólitos; es decir, un resultado que difiere de lo que se espera en una comparación carácter a carácter. Los ejemplos que se muestran suponen que se utiliza la misma página de códigos MBCS, sin embargo, la situación puede complicarse más cuando se utilizan distintas páginas de códigos de múltiples bytes con el mismo idioma nacional. Por ejemplo,

considere el caso de la comparación de una serie de una página de códigos DBCS japonesa y una página de códigos EUC japonesa.

Comparaciones de indicación de fecha y hora

Un valor DATE, TIME o TIMESTAMP puede compararse con otro valor del mismo tipo de datos o con una representación de serie de dicho tipo de datos. Todas las comparaciones son cronológicas, lo que significa que cuanto más alejado en el tiempo esté del 1 de enero de 0001, mayor es el valor de dicho punto en el tiempo.

Las comparaciones que implican valores TIME y representaciones de serie de valores de hora siempre incluyen los segundos. Si la representación de serie omite los segundos, se implica que son cero segundos.

Las comparaciones que implican valores TIMESTAMP son cronológicas sin tener en cuenta las representaciones que puedan considerarse equivalentes.

Ejemplo:

```
TIMESTAMP('1990-02-23-00.00.00') > '1990-02-22-24.00.00'
```

Comparaciones de tipos definidos por el usuario

Los valores con un tipo diferenciado definido por el usuario sólo se pueden comparar con valores que sean exactamente del mismo tipo diferenciado definido por el usuario. El tipo diferenciado definido por el usuario debe haberse definido utilizando la cláusula WITH COMPARISONS.

Ejemplo:

Considere por ejemplo el siguiente tipo diferenciado YOUTH y la tabla CAMP_DB2_ROSTER:

```
CREATE DISTINCT TYPE YOUTH AS INTEGER WITH COMPARISONS
```

```
CREATE TABLE CAMP_DB2_ROSTER
( NAME          VARCHAR(20),
  ATTENDEE_NUMBER INTEGER NOT NULL,
  AGE           YOUTH,
  HIGH_SCHOOL_LEVEL YOUTH)
```

La comparación siguiente es válida:

```
SELECT * FROM CAMP_DB2_ROSTER
WHERE AGE > HIGH_SCHOOL_LEVEL
```

La comparación siguiente no es válida:

```
SELECT * FROM CAMP_DB2_ROSTER
WHERE AGE > ATTENDEE_NUMBER
```

Asignaciones y comparaciones

Sin embargo, AGE se puede comparar con ATTENDEE_NUMBER utilizando una función o una especificación CAST para convertir entre el tipo diferenciado y el tipo fuente. Todas las comparaciones siguientes son válidas:

```
SELECT * FROM CAMP_DB2_ROSTER
WHERE INTEGER(AGE) > ATTENDEE_NUMBER
```

```
SELECT * FROM CAMP_DB2_ROSTER
WHERE CAST( AGE AS INTEGER) > ATTENDEE_NUMBER
```

```
SELECT * FROM CAMP_DB2_ROSTER
WHERE AGE > YOUTH(ATTENDEE_NUMBER)
```

```
SELECT * FROM CAMP_DB2_ROSTER
WHERE AGE > CAST(ATTENDEE_NUMBER AS YOUTH)
```

Los valores con un tipo estructurado definido por el usuario no se pueden comparar con ningún otro valor (se pueden utilizar los predicados NULL y TYPE).

Comparaciones de tipos de referencia

Los valores de un tipo de referencia sólo pueden compararse si sus tipos de destino tienen un supertipo común. Sólo se encontrará la función de comparación adecuada si el nombre de esquema del supertipo común se incluye en la vía de acceso de la función. Se realiza la comparación si se utiliza el tipo de representación de los tipos de referencia. El ámbito de la referencia no se tiene en cuenta en la comparación.

Reglas para los tipos de datos del resultado

Los tipos de datos de un resultado los determinan las reglas que se aplican a los operandos de una operación. Esta sección explica dichas reglas.

Estas reglas se aplican a:

- Las columnas correspondientes en selecciones completas de operaciones de conjuntos (UNION, INTERSECT y EXCEPT)
- Expresiones resultantes de una expresión CASE
- Los argumentos de la función escalar COALESCE (o VALUE)
- Los valores de expresiones contenidos en la lista de entrada de un predicado IN
- Las expresiones correspondientes de una cláusula VALUES de múltiples filas.

Estas reglas se aplican, sujetas a otras restricciones, sobre series largas para las distintas operaciones.

Reglas para los tipos de datos del resultado

A continuación encontrará las reglas que se refieren a los distintos tipos de datos. En algunos casos, se utiliza una tabla para mostrar los posibles tipos de datos resultantes.

Estas tablas identifican el tipo de datos resultante, incluida la longitud aplicable o precisión y la escala. El tipo resultante se determina teniendo en cuenta los operandos. Si hay más de un par de operandos, empiece considerando el primer par. Esto da un tipo resultante que es el que se examina con el siguiente operando para determinar el siguiente tipo resultante, etcétera. El último tipo resultante intermedio y el último operando determinan el tipo resultante para la operación. El proceso de operaciones se realiza de izquierda a derecha, por lo tanto los tipos del resultado intermedios son importantes cuando se repiten operaciones. Por ejemplo, examinemos una situación que implique:

```
CHAR(2) UNION CHAR(4) UNION VARCHAR(3)
```

El primer par da como resultado un tipo CHAR(4). Los valores del resultado siempre tienen 4 caracteres. El tipo resultante final es VARCHAR(4). Los valores del resultado de la primera operación UNION siempre tendrán una longitud de 4.

Series de caracteres

Las series de caracteres son compatibles con otras series de caracteres. Las series de caracteres incluyen los tipos CHAR, VARCHAR, LONG VARCHAR y CLOB.

Si un operando es...	Y el otro operando es...	El tipo de datos del resultado es...
CHAR(x)	CHAR(y)	CHAR(z) donde $z = \max(x,y)$
CHAR(x)	VARCHAR(y)	VARCHAR(z) donde $z = \max(x,y)$
VARCHAR(x)	CHAR(y) o VARCHAR(y)	VARCHAR(z) donde $z = \max(x,y)$
LONG VARCHAR	CHAR(y), VARCHAR(y) o LONG VARCHAR	LONG VARCHAR
CLOB(x)	CHAR(y), VARCHAR(y) o CLOB(y)	CLOB(z) donde $z = \max(x,y)$
CLOB(x)	LONG VARCHAR	CLOB(z) donde $z = \max(x,32700)$

La página de códigos de la serie de caracteres del resultado se derivará basándose en el apartado “Reglas para las conversiones de series” en la página 124.

Reglas para los tipos de datos del resultado

Series gráficas

Las series gráficas son compatibles con otras series gráficas. Las series gráficas incluyen los tipos de datos GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC y DBCLOB.

Si un operando es...	Y el otro operando es...	El tipo de datos del resultado es...
GRAPHIC(x)	GRAPHIC(y)	GRAPHIC(z) donde $z = \max(x,y)$
VARGRAPHIC(x)	GRAPHIC(y) o VARGRAPHIC(y)	VARGRAPHIC(z) donde $z = \max(x,y)$
LONG VARGRAPHIC	GRAPHIC(y), VARGRAPHIC(y) o LONG VARGRAPHIC	LONG VARGRAPHIC
DBCLOB(x)	GRAPHIC(y), VARGRAPHIC(y) o DBCLOB(y)	DBCLOB(z) donde $z = \max(x,y)$
DBCLOB(x)	LONG VARGRAPHIC	DBCLOB(z) donde $z = \max(x,16350)$

La página de códigos de la serie gráfica del resultado se derivará basándose en el apartado “Reglas para las conversiones de series” en la página 124.

Gran objeto binario (BLOB)

Un BLOB sólo es compatible con otro BLOB y el resultado es un BLOB. Debe utilizarse la función escalar BLOB para convertir desde otros tipos si deben tratarse como tipos BLOB (vea “BLOB” en la página 287). La longitud del resultado BLOB es la longitud mayor de todos los tipos de datos.

Numérico

Los tipos numéricos son compatibles con otros tipos numéricos. Los tipos numéricos incluyen SMALLINT, INTEGER, BIGINT, DECIMAL, REAL y DOUBLE.

Si un operando es...	Y el otro operando es...	El tipo de datos del resultado es...
SMALLINT	SMALLINT	SMALLINT
INTEGER	INTEGER	INTEGER
INTEGER	SMALLINT	INTEGER
BIGINT	BIGINT	BIGINT
BIGINT	INTEGER	BIGINT
BIGINT	SMALLINT	BIGINT
DECIMAL(w,x)	SMALLINT	DECIMAL(p,x) donde $p = x + \max(w-x, 5)$ ¹

Reglas para los tipos de datos del resultado

Si un operando es...	Y el otro operando es...	El tipo de datos del resultado es...
DECIMAL(w,x)	INTEGER	DECIMAL(p,x) donde $p = x + \max(w-x, 11)^1$
DECIMAL(w,x)	BIGINT	DECIMAL(p,x) donde $p = x + \max(w-x, 19)^1$
DECIMAL(w,x)	DECIMAL(y,z)	DECIMAL(p,s) donde $p = \max(x,z) + \max(w-x, y-z)^1$ $s = \max(x,z)$
REAL	REAL	REAL
REAL	DECIMAL, BIGINT, INTEGER o SMALLINT	DOUBLE
DOUBLE	cualquier tipo numérico	DOUBLE

Nota: 1. La precisión no puede exceder de 31.

DATE

Una fecha es compatible con otra fecha o con cualquier expresión CHAR o VARCHAR que contiene una representación de serie válida de una fecha. El tipo de datos del resultado es DATE.

TIME

Una hora es compatible con otra hora o con cualquier expresión CHAR o VARCHAR que contenga una representación de serie válida de una hora. El tipo de datos del resultado es TIME.

TIMESTAMP

Una indicación de la hora es compatible con otra indicación de la hora o con cualquier expresión CHAR o VARCHAR que contenga una representación de serie válida de una indicación de la hora. El tipo de datos del resultado es TIMESTAMP.

DATALINK

Un enlace de datos es compatible con otro enlace de datos. El tipo de datos del resultado es DATALINK. La longitud del resultado DATALINK es la longitud mayor de todos los tipos de datos.

Tipos definidos por el usuario

Tipos diferenciados

Un tipo diferenciado definido por el usuario sólo es compatible con el mismo tipo diferenciado definido por el usuario. El tipo de datos del resultado es el tipo diferenciado definido por el usuario.

Tipos de referencia

Un tipo de referencia es compatible con otro tipo de referencia en el caso de que sus tipos de destino tengan un supertipo común. El tipo de datos del

Reglas para los tipos de datos del resultado

resultado es un tipo de referencia que tiene un supertipo común como tipo de destino. Si todos los operandos tienen la tabla de ámbito idéntica, el resultado tiene esta tabla de ámbito. De lo contrario, el resultado no tiene ámbito.

Tipos estructurados

Un tipo estructurado es compatible con otro tipo de estructurado siempre tengan un supertipo común. El tipo de datos estático de la columna del tipo estructurado resultante es el tipo estructurado que es el supertipo menos común de cualquiera de las dos columnas.

Por ejemplo, considere la siguiente jerarquía de tipos estructurados:



Los tipos estructurados del tipo estático E y F son compatibles con el tipo estático resultante de B, que es el supertipo menos común de E y F.

Atributo de posibilidad de nulo del resultado

A excepción de INTERSECT y EXCEPT, el resultado permite nulos a menos que ambos operandos no permitan nulos.

- Para INTERSECT, si cualquier operando no permite nulos, el resultado no permite nulos (la intersección nunca sería nula).
- Para EXCEPT, si el primer operando no permite nulos, el resultado no permite nulos (el resultado sólo puede ser valores del primer operando).

Reglas para las conversiones de series

La página de códigos utilizada para realizar una operación se determina por las reglas que se aplican a los operandos en dicha operación. Esta sección explica dichas reglas.

Estas reglas se aplican a:

- Las columnas de serie correspondientes en selecciones completas con operaciones de conjuntos (UNION, INTERSECT y EXCEPT)
- Los operandos de concatenación
- Los operandos de predicados (a excepción de LIKE)
- Expresiones resultantes de una expresión CASE
- Los argumentos de la función escalar COALESCE (y VALUE)
- Los valores de expresiones de la lista de entrada de un predicado IN
- Las expresiones correspondientes de la cláusula VALUES de múltiples filas.

Reglas para las conversiones de series

En cada caso, la página de códigos del resultado se determina en el momento del enlace, y la ejecución de la operación puede implicar la conversión de series a la página de códigos identificada por dicha página de códigos. Un carácter que no tenga una conversión válida se correlaciona con el carácter de sustitución del juego de caracteres y SQLWARN10 se establece en 'W' en la SQLCA.

La página de códigos del resultado se determina por las páginas de códigos de los operandos. Las páginas de códigos de los dos primeros operandos determinan una página de códigos del resultado intermedia, esta página de códigos y la del siguiente operando determinan una nueva página de códigos del resultado intermedia (si se aplica), etcétera. La última página de códigos del resultado intermedia y la página de códigos del último operando determinan la página de códigos de la serie o columna del resultado. En cada par de páginas de códigos, el resultado se determina por la aplicación secuencial de las reglas siguientes:

- Si las páginas de códigos son iguales, el resultado es dicha página de códigos.
- Si cualquiera de las dos páginas de códigos es BIT DATA (página de códigos 0), la página de códigos del resultado es BIT DATA.
- De lo contrario, la página de códigos del resultado se determina por la Tabla 8. Una entrada 'primer' en la tabla significa que se selecciona la página de códigos del primer operando y una entrada 'segundo' significa que se selecciona la página de códigos del segundo operando.

Tabla 8. Selección de la página de códigos del resultado intermedio

Primer operando	Segundo operando				
	Valor de columna	Valor derivado	Constante	Registro especial	Variable del lenguaje principal
Valor de columna	primer	primer	primer	primer	primer
Valor derivado	segundo	primer	primer	primer	primer
Constante	segundo	segundo	primer	primer	primer
Registro especial	segundo	segundo	primer	primer	primer
Variable del lenguaje principal	segundo	segundo	segundo	segundo	primer

Un resultado intermedio se considera un operando de valor derivado. Una expresión que no sea un solo valor de columna, constante, registro especial ni

Reglas para las conversiones de series

variable del lenguaje principal también se considera un operando de valor derivado. Existe una excepción a esta regla si la expresión es una especificación CAST (o una llamada a una función que sea equivalente). En este caso, la *clase* del primer operando se basa en el primer argumento de la especificación CAST.

Se considera que las columnas de vista tienen el tipo de operando del objeto en el que están basadas en última instancia. Por ejemplo, una columna de vista definida en una columna de tabla se considera un valor de columna, mientras que una columna de vista basada en una expresión de serie (por ejemplo, A CONCAT B) se considera un valor derivado.

Las conversiones a la página de códigos del resultado se realizan, si es necesario, para:

- Un operando del operador de concatenación
- El argumento seleccionado de la función escalar COALESCE (o VALUE)
- La expresión del resultado seleccionada de la expresión CASE
- Las expresiones de la lista in del predicado IN
- Las expresiones correspondientes de una cláusula VALUES de múltiples filas
- Las columnas correspondientes que hacen referencia en operaciones de conjuntos.

La conversión de los caracteres es necesaria si son ciertas todas las afirmaciones siguientes:

- Las páginas de códigos son diferentes
- Ninguna serie es BIT DATA
- La serie no es nula ni está vacía
- La tabla de selección de conversión de página de códigos indica que es necesaria la conversión.

Ejemplos

Ejemplo 1: Suponga lo siguiente:

Expresión	Tipo	Página de códigos
COL_1	columna	850
HV_2	variable del lenguaje principal	437

Cuando se evalúa el predicado:

COL_1 CONCAT :HV_2

La página de códigos del resultado de los dos operandos es 850, ya que el operando dominante es la columna COL_1.

Ejemplo 2: Utilizando la información del ejemplo anterior, cuando se evalúa el predicado:

```
COALESCE(COL_1, :HV_2:NULLIND,)
```

La página de códigos del resultado es 850. Por lo tanto, la página de códigos del resultado para la función escalar COALESCE será la página de códigos 850.

Compatibilidad entre particiones

La *compatibilidad entre particiones* se define entre los tipos de base de datos de las columnas correspondientes de las claves de particionamiento. Los tipos de datos compatibles de partición tienen la propiedad de que dos variables, una de cada tipo, con el mismo valor, se correlacionan con el mismo índice de mapa de particionamiento por la misma función de partición.

La Tabla 9 en la página 128 muestra la compatibilidad de los tipos de datos en particiones.

La compatibilidad entre particiones tiene las características siguientes:

- Se utilizan formatos internos para DATE, TIME y TIMESTAMP. No son compatibles entre sí y ninguno es compatible con CHAR.
- La compatibilidad entre particiones no se ve afectada por las columnas con definiciones NOT NULL o FOR BIT DATA.
- Los valores NULL de los tipos de datos compatibles se tratan de manera idéntica. Se pueden generar resultados diferentes para los valores NULL de tipos de datos no compatibles.
- Se utiliza el tipo de datos base del UDT para analizar la compatibilidad entre particiones.
- Los decimales del mismo valor de la clave de particionamiento se tratan de manera idéntica, incluso si difieren su escala y precisión.
- La función de generación aleatoria proporcionada por el sistema ignora los blancos de cola de las series de caracteres (CHAR, VARCHAR, GRAPHIC o VARGRAPHIC).
- CHAR o VARCHAR de diferentes longitudes son tipos de datos compatibles.
- Los valores REAL o DOUBLE se tratan de manera idéntica incluso si su precisión es diferente.

Compatibilidad entre particiones

Tabla 9. Compatibilidades entre particiones

Operandos	Entero binario	Número decimal	Coma flotante	Serie de caracteres	Serie gráfica	Fecha	Hora	Indicación de fecha y hora	Tipo diferenciado	Tipo estructurado
Entero binario	Sí	No	No	No	No	No	No	No	¹	No
Número decimal	No	Sí	No	No	No	No	No	No	¹	No
Coma flotante	No	No	Sí	No	No	No	No	No	¹	No
Serie de caracteres ³	No	No	No	Sí ²	No	No	No	No	¹	No
Serie gráfica ³	No	No	No	No	Sí	No	No	No	¹	No
Fecha	No	No	No	No	No	Sí	No	No	¹	No
Hora	No	No	No	No	No	No	Sí	No	¹	No
Indicación de fecha y hora	No	No	No	No	No	No	No	Sí	¹	No
Tipo diferenciado	1	1	1	1	1	1	1	1	1	No
Tipo estructurado ³	No	No	No	No	No	No	No	No	No	No

Nota:

- ¹ El valor de un tipo diferenciado definido por el usuario (UDT) es compatible, a nivel de partición, con el tipo fuente del UDT o cualquier otro UDT con un tipo fuente compatible a nivel de partición.
- ² El atributo FOR BIT DATA no afecta a la compatibilidad entre particiones.
- ³ Observe que los tipos estructurados definidos por el usuario y los tipos de datos LONG VARCHAR, LONG VARGRAPHIC, CLOB, DBCLOB y BLOB no son aplicables para la compatibilidad de particiones, pues no están soportados en las claves de particionamiento.

Constantes

Una *constante* (a veces llamada un *literal*) especifica un valor. Las constantes se clasifican en constantes de tipo serie y constantes numéricas. Las constantes numéricas pueden, a su vez, ser constantes enteras, de coma flotante y decimales.

Todas las constantes tienen el atributo NOT NULL.

Un valor de cero negativo en una constante numérica (-0) indica el mismo valor que un cero sin el signo (0).

Constantes enteras

Una *constante entera* especifica un entero en forma de número, con signo o sin signo, con un máximo de 19 dígitos que no incluye ninguna coma decimal. El tipo de datos de una constante entera es un entero grande si su valor está comprendido en el rango de un entero grande. El tipo de datos de una constante entera es un entero grande si su valor se encuentra fuera del rango de un entero grande, pero está comprendido en el rango de un entero grande. Una constante definida fuera del rango de valores enteros superior se considera una constante decimal.

Observe que la representación literal más pequeña de una constante entera grande es -2.147.483.647 y no -2.147.483.648, que es el límite para los valores enteros. De manera similar, la representación literal más pequeña de una constante entera grande es -9.223.372.036.854.775.807 y no -9.223.372.036.854.775.;808, que es el límite para los valores enteros grandes.

Ejemplos

64 -15 +100 32767 720176 12345678901

En los diagramas de sintaxis, el término 'entero' se utiliza para una constante entera grande que no debe incluir un signo.

Constantes de coma flotante

Una *constante de coma flotante* especifica un número de coma flotante en forma de dos números separados por una E. El primer número puede incluir un signo y una coma decimal; el segundo número puede incluir un signo, pero no una coma decimal. El tipo de datos de una constante de coma flotante es de precisión doble. El valor de la constante es el producto del primer número y la potencia de 10 especificada por el segundo número; este valor debe estar dentro del rango de los números de coma flotante. El número de caracteres de la constante no debe exceder de 30.

Ejemplos

15E1 2,E5 2,2E-1 +5,E+2

Constantes

Constantes decimales

Una *constante decimal* es un número con o sin signo de 31 dígitos de longitud como máximo y que incluye una coma decimal o no está comprendido dentro del rango de enteros binarios. Debe encontrarse en el rango de números decimales. La precisión es el número total de dígitos (incluyendo los ceros iniciales y de cola); la escala es el número de dígitos situados a la derecha de la coma decimal (incluyendo los ceros de cola).

Ejemplos

25,5 1000, -15, +37589,3333333333

Constantes de tipo serie

Una *constante de tipo serie* especifica una serie de caracteres de longitud variable y consta de una secuencia de caracteres que empieza y finaliza por un apóstrofo ('). Esta modalidad de constante de tipo serie especifica la serie de caracteres contenida entre los delimitadores de series. La longitud de la serie de caracteres no debe ser mayor que 32.672 bytes. Se utilizan dos delimitadores de series consecutivos para representar un delimitador de series dentro de la serie de caracteres.

Ejemplos

```
'14/12/1985'  
'32'  
'DON''T CHANGE'
```

Consideraciones acerca de páginas de códigos diferentes

El valor de una constante se convierte siempre en la página de códigos de la base de datos cuando se enlaza con la base de datos. Se considera que está en la página de códigos de la base de datos. Por lo tanto, si se utiliza en una expresión que combina una constante con una columna FOR BIT DATA, cuyo resultado es FOR BIT DATA, el valor de la constante *no* se convertirá desde su representación de página de códigos de base de datos.

Constantes hexadecimales

Una *constante hexadecimal* especifica una serie de caracteres de longitud variable con la página de códigos del servidor de aplicaciones.

El formato de una constante de serie hexadecimal es una X seguida por una secuencia de caracteres que empieza y termina con un apóstrofo. Los caracteres que se incluyen entre los apóstrofes deben corresponder a una cantidad par de dígitos hexadecimales. El número de dígitos hexadecimales no debe exceder de 16.336 o, de lo contrario, se producirá un error (SQLSTATE -54002). Un dígito hexadecimal representa 4 bits. Si se especifica como un dígito o cualquiera de las letras de la A a la F (mayúsculas o minúsculas) donde A representa el patrón de bits '1010', B el patrón de bits '1011', etc. Si una constante hexadecimal no tiene el formato correcto (p. ej.,

contiene un dígito hexadecimal no válido o un número impar de dígitos hexadecimales), se genera un error (SQLSTATE 42606).

Ejemplos

X'FFFF' representa el patrón de bits '1111111111111111'

X'4672616E6B' representa el patrón VARCHAR de la serie ASCII 'Frank'

Constantes gráficas de tipo serie

Una *constante gráfica de tipo serie* especifica una serie gráfica de longitud variable y consta de una secuencia de caracteres de doble byte que empieza y termina con un apóstrofo de un solo byte ('), y está precedida por una G o una N de un solo byte. La longitud de la serie gráfica debe ser un número par de bytes y no debe exceder de 16.336 bytes.

Ejemplos:

G'serie de caracteres de doble byte'
N'serie de caracteres de doble byte'

Consideraciones acerca de MBCS

El apóstrofo no debe aparecer como parte de un carácter MBCS para que se considere como delimitador.

Utilización de constantes con tipos definidos por el usuario

Los tipos definidos por el usuario son difíciles de escribir. Esto significa que un tipo definido por el usuario sólo es compatible con su propio tipo. Sin embargo, una constante tiene un tipo incorporado. Por lo tanto, una operación que implique un tipo definido por el usuario y una constante sólo es posible si el tipo definido por el usuario se ha convertido al tipo interno de la constante o la constante se ha convertido al tipo definido por el usuario (vea "Especificaciones CAST" en la página 193 para obtener información sobre la conversión de tipos de datos). Por ejemplo, si se utiliza la tabla y el tipo diferenciado del apartado "Comparaciones de tipos definidos por el usuario" en la página 119, serán válidas las siguientes comparaciones con la constante 14:

```
SELECT * FROM CAMP_DB2_ROSTER
  WHERE AGE > CAST(14 AS YOUTH)

SELECT * FROM CAMP_DB2_ROSTER
  WHERE CAST(AGE AS INTEGER) > 14
```

No será válida la siguiente comparación:

```
SELECT * FROM CAMP_DB2_ROSTER
  WHERE AGE > 14
```

Registros especiales

Registros especiales

Un *registro especial* es un área de almacenamiento que un gestor de bases de datos define para un proceso de aplicación y que sirve para almacenar información a la que se puede hacer referencia en sentencias de SQL. Los registros especiales se encuentran en la página de códigos de la base de datos.

CLIENT ACCTNG

El registro especial CLIENT ACCTNG contiene el valor de la serie de contabilidad de la información de cliente especificada para esta conexión. El tipo de datos del registro es VARCHAR(255). El valor por omisión de este registro es una serie vacía.

El valor de la serie de contabilidad puede cambiarse utilizando la API para Establecer información de cliente (sqleseti).

Tenga en cuenta que el valor proporcionado a través de la API sqleseti está en la página de códigos de aplicación y el valor del registro especial se almacena en la página de códigos de base de datos. En función de los valores de datos utilizados al establecer la información de cliente, puede que, durante la conversión de la página de códigos, se produzca el truncamiento del valor de datos almacenado en el registro especial.

Ejemplo

Obtenga el valor actual de la serie de contabilidad para esta conexión.

```
VALUES (CLIENT ACCTNG)
INTO :ACCT_STRING
```

CLIENT APPLNAME

El registro especial CLIENT APPLNAME contiene el valor del nombre de aplicación de la información de cliente especificada para esta conexión. El tipo de datos del registro es VARCHAR(255). El valor por omisión de este registro es una serie vacía.

El valor del nombre de aplicación puede cambiarse utilizando la API para Establecer información de cliente (sqleseti).

Tenga en cuenta que el valor proporcionado a través de la API sqleseti está en la página de códigos de aplicación y el valor del registro especial se almacena en la página de códigos de base de datos. En función de los valores de datos utilizados al establecer la información de cliente, puede que, durante la conversión de la página de códigos, se produzca el truncamiento del valor de datos almacenado en el registro especial.

Ejemplo

Seleccione los departamentos a los que se les permite utilizar la aplicación que se está usando en esta conexión.

```
SELECT DEPT
      FROM DEPT_APPL_MAP
      WHERE APPL_NAME = CLIENT APPLNAME
```

CLIENT USERID

El registro especial CLIENT USERID contiene el valor del ID de usuario de cliente de la información de cliente especificada para esta conexión. El tipo de datos del registro es VARCHAR(255). El valor por omisión de este registro es una serie vacía.

El valor del ID de usuario de cliente puede cambiarse utilizando la API para Establecer información de cliente (sqleseti).

Tenga en cuenta que el valor proporcionado a través de la API sqleseti está en la página de códigos de aplicación y el valor del registro especial se almacena en la página de códigos de base de datos. En función de los valores de datos utilizados al establecer la información de cliente, puede que, durante la conversión de la página de códigos, se produzca el truncamiento del valor de datos almacenado en el registro especial.

Ejemplo

Averigüe en qué departamento funciona el ID de usuario de cliente actual.

```
SELECT DEPT
      FROM DEPT_USERID_MAP
      WHERE USER_ID = CLIENT USERID
```

CLIENT WRKSTNNAME

El registro especial CLIENT WRKSTNNAME contiene el valor del nombre de estación de trabajo de la información de cliente especificada para esta conexión. El tipo de datos del registro es VARCHAR(255). El valor por omisión de este registro es una serie vacía.

El valor del nombre de estación de trabajo puede cambiarse utilizando la API para Establecer información de cliente (sqleseti).

Tenga en cuenta que el valor proporcionado a través de la API sqleseti está en la página de códigos de aplicación y el valor del registro especial se almacena en la página de códigos de base de datos. En función de los valores de datos utilizados al establecer la información de cliente, puede que, durante la conversión de la página de códigos, se produzca el truncamiento del valor de datos almacenado en el registro especial.

Ejemplo

Obtenga el nombre de estación de trabajo que se está utilizando para esta conexión.

```
VALUES (CLIENT WRKSTNNAME)
      INTO :WS_NAME
```

Registros especiales

CURRENT DATE

El registro especial CURRENT DATE especifica una fecha basada en la lectura del reloj cuando se ejecuta una sentencia de SQL en el servidor de aplicación. Si este registro especial se utiliza más de una vez en la misma sentencia de SQL o bien con CURRENT TIME o CURRENT TIMESTAMP en una sola sentencia, todos los valores se basan en la misma lectura del reloj.

En un sistema federado, CURRENT DATE se puede utilizar en una consulta prevista para fuentes de datos. Cuando se procesa la consulta, la fecha devuelta se obtendrá del registro CURRENT DATE, en el servidor federado, no de las fuentes de datos.

Ejemplo

Utilizando la tablas PROJECT, establezca la fecha final del proyecto (PRENDATE) del proyecto MA2111 (PROJNO) en la fecha actual.

```
UPDATE PROJECT
  SET PRENDATE = CURRENT DATE
  WHERE PROJNO = 'MA2111'
```

CURRENT DEFAULT TRANSFORM GROUP

El registro especial CURRENT DEFAULT TRANSFORM GROUP especifica un valor VARCHAR (18) que identifica el nombre del grupo de transformación utilizado por las sentencias de SQL dinámicas para intercambiar valores de tipos estructurados, definidos por el usuario, con programas de lenguaje principal. Este registro especial no especifica los grupos de transformación utilizados en las sentencias de SQL dinámicas o en el intercambio de parámetros y resultados con funciones externas de métodos.

Su valor puede definirse mediante la sentencia SET CURRENT DEFAULT TRANSFORM GROUP. Si no se define ningún valor, el valor inicial del registro especial es la serie vacía (valor VARCHAR de longitud cero).

En una sentencia de SQL dinámica (es decir, una sentencia que interactúa con variables del lenguaje principal), el nombre del grupo de transformación utilizado para intercambiar valores es el mismo que el nombre de este registro especial, a menos que el registro contenga la serie vacía. Si el registro contiene la serie vacía (no se ha definido ningún valor utilizando la sentencia SET CURRENT DEFAULT TRANSFORM GROUP), se utiliza el grupo de transformación DB2_PROGRAM para la transformación. Si el grupo de transformación DB2_PROGRAM no está definido para el tipo estructurado indicado, se emite un error durante la ejecución (SQLSTATE 42741).

Ejemplo

Establezca el grupo de transformación por omisión en MYSTRUCT1. Las funciones TO SQL y FROM SQL definidas en la transformación MYSTRUCT1

se utilizan para intercambiar variables de tipo estructurado, definidas por el usuario, con el programa de lenguaje principal.

```
SET CURRENT DEFAULT TRANSFORM GROUP = MYSTRUCT1
```

Recupere el nombre del grupo de transformación por omisión asignado a este registro especial.

```
VALUES (CURRENT DEFAULT TRANSFORM GROUP)
```

CURRENT DEGREE

El registro especial CURRENT DEGREE especifica los grados de paralelismo intrapartición para la ejecución de sentencias de SQL dinámicas.¹³ El tipo de datos del registro es CHAR(5). Los valores válidos son 'ANY' o la representación de serie de un entero entre 1 y 32 767, inclusive.

Si el valor de CURRENT DEGREE representado como un entero es 1 cuando la sentencia de SQL se prepara dinámicamente, la ejecución de esta sentencia no utilizará el paralelismo intrapartición.

Si el valor de CURRENT DEGREE representado como un entero es mayor que 1 y menor o igual a 32 767 cuando una sentencia de SQL se prepara dinámicamente, la ejecución de esta sentencia puede implicar el paralelismo intrapartición con el grado especificado.

Si el valor de CURRENT DEGREE es 'ANY' cuando una sentencia de SQL se prepara dinámicamente, la ejecución de dicha sentencia puede implicar el paralelismo intrapartición que utiliza un grado determinado por el gestor de bases de datos.

El grado de paralelismo real durante la ejecución será el menor de los valores siguientes:

- El parámetro de configuración de grado máximo de consulta (max_querydegree)
- El grado del tiempo de ejecución de la aplicación
- El grado de compilación de la sentencia de SQL

Si el parámetro de configuración del gestor de bases de datos, intra_parallel, se establece en NO, el valor del registro especial CURRENT DEGREE se ignorará con el fin de optimizar y la sentencia no utilizará el paralelismo intrapartición.

Consulte el manual *Administration Guide* para ver una descripción del paralelismo y una lista de restricciones.

13. En el caso de SQL estático, la opción de enlace lógico DEGREE proporciona el mismo control.

Registros especiales

Se puede cambiar el valor ejecutando la sentencia SET CURRENT DEGREE (consulte el apartado “SET CURRENT DEGREE” en la página 1151 para obtener información sobre esta sentencia).

El valor inicial de CURRENT DEGREE lo determina el parámetro de configuración de la base de datos dft_degree. Consulte el manual *Administration Guide* para ver una descripción de este parámetro de configuración.

CURRENT EXPLAIN MODE

El registro especial CURRENT EXPLAIN MODE contiene un valor VARCHAR(254) que controla la actuación del recurso Explain con respecto a sentencias de SQL dinámicas admisibles. Este recurso genera e inserta información de Explain en las tablas de Explain (para obtener más información, vea el manual *Administration Guide*). Esta información no incluye la instantánea de Explain.

Los valores posibles son YES, NO, EXPLAIN, RECOMMEND INDEXES y EVALUATE INDEXES.¹⁴

YES Habilita el recurso Explain y hace que la información de Explain para una sentencia de SQL dinámica se capture al compilar la sentencia.

EXPLAIN

Habilita el recurso igual que YES, sin embargo, no se ejecutan las sentencias dinámicas.

NO Inhabilita el recurso Explain.

RECOMMEND INDEXES

Para cada consulta dinámica, se recomienda un conjunto de índices. La tabla ADVISE_INDEX se llena con el conjunto de índices.

EVALUATE INDEXES

Las consultas dinámicas se explican como si existieran los índices recomendados. Los índices se seleccionan de la tabla ADVISE_INDEX.

El valor inicial es NO.

La sentencia SET CURRENT EXPLAIN MODE puede cambiar su valor (consulte el apartado “SET CURRENT EXPLAIN MODE” en la página 1153 para obtener información sobre esta sentencia).

Los valores de los registros especiales CURRENT EXPLAIN MODE y CURRENT EXPLAIN SNAPSHOT interactúan al invocar el recurso Explain

14. En el caso del SQL estático, la opción de enlace de EXPLAIN proporciona el mismo control. En el caso de los mandatos PREP y BIND, los valores de la opción EXPLAIN son: YES, NO y ALL.

(consulte la Tabla 145 en la página 1495 obtener para más detalles). El registro especial CURRENT EXPLAIN MODE también interactúa con la opción de enlace EXPLAIN (consulte la Tabla 146 en la página 1496 para obtener más detalles). Los valores RECOMMEND INDEXES y EVALUATE INDEXES sólo se pueden establecer para el registro CURRENT EXPLAIN MODE y se han de establecer utilizando la sentencia SET CURRENT EXPLAIN MODE.

Ejemplo: Establezca la variable de lenguaje principal EXPL_MODE (VARCHAR(254)) en el valor que hay actualmente en el registro especial CURRENT EXPLAIN MODE.

```
VALUES CURRENT EXPLAIN MODE
INTO :EXPL_MODE
```

CURRENT EXPLAIN SNAPSHOT

El registro especial CURRENT EXPLAIN SNAPSHOT contiene un valor CHAR(8) que controla el funcionamiento del recurso de instantáneas de Explain. Este recurso genera información comprimida que incluye información sobre planes de acceso, costes del operador y estadísticas en tiempo de enlace lógico (para más información, consulte el manual *Administration Guide*).

Sólo las siguientes sentencias tienen en cuenta el valor de este registro: DELETE, INSERT, SELECT, SELECT INTO, UPDATE, VALUES y VALUES INTO.

Los valores posibles son YES, NO y EXPLAIN.¹⁵

YES Habilita el recurso de instantáneas y toma una instantánea de la representación interna de una sentencia de SQL dinámica al compilar la sentencia.

EXPLAIN

Habilita el recurso igual que YES, sin embargo, no se ejecutan las sentencias dinámicas.

NO Inhabilita el servicio de instantánea de Explain.

El valor inicial es NO.

La sentencia SET CURRENT EXPLAIN SNAPSHOT puede cambiar este valor (consulte el apartado “SET CURRENT EXPLAIN SNAPSHOT” en la página 1155).

Los valores de los registros especiales CURRENT EXPLAIN SNAPSHOT y CURRENT EXPLAIN MODE interactúan al invocar el recurso Explain

15. En el caso del SQL estático, la opción de enlace lógico EXPLSNAP proporciona el mismo control. En el caso de los mandatos PREP y BIND, los valores de la opción EXPLSNAP son: YES, NO y ALL.

Registros especiales

(consulte la Tabla 145 en la página 1495 para más detalles). El registro especial CURRENT EXPLAIN SNAPSHOT también interactúa con la opción de enlace EXPLSNAP (consulte la Tabla 147 en la página 1498 para más detalles).

Ejemplo

Establezca la variable del lenguaje principal EXPL_SNAP (char(8)) en el valor que contiene actualmente el registro especial CURRENT EXPLAIN SNAPSHOT.

```
VALUES CURRENT EXPLAIN SNAPSHOT
INTO :EXPL_SNAP
```

CURRENT NODE

El registro especial CURRENT NODE especifica un valor INTEGER que identifica el número de nodo coordinador (la partición a la que se conecta la aplicación).

CURRENT NODE devuelve 0 si la instancia de la base de datos no está definida para dar soporte a la partición (no existe ningún archivo db2nodes.cfg¹⁶).

La sentencia CONNECT puede cambiar CURRENT NODE solamente bajo determinadas circunstancias (consulte el apartado “CONNECT (Tipo 1)” en la página 653).

Ejemplo

Establezca la variable del lenguaje principal APPL_NODE (entero) en el número de la partición a la que está conectada la aplicación.

```
VALUES CURRENT NODE
INTO :APPL_NODE
```

CURRENT PATH

El registro especial CURRENT PATH especifica un valor VARCHAR(254) que identifica la *vía de acceso de SQL* que se ha de utilizar para resolver las referencias de funciones y las referencias de tipos de datos que se utilizan en sentencias de SQL preparadas dinámicamente.¹⁷ CURRENT PATH también se utiliza para resolver las referencias de procedimientos almacenados en sentencias CALL. El valor inicial es el valor por omisión que se especifica más abajo. Para el SQL estático, la opción de enlace lógico FUNCPATH proporciona una vía de acceso de SQL que sirve para la resolución de funciones y tipos de datos (consulte la publicación *Consulta de mandatos* para obtener más información sobre la opción de enlace FUNCPATH).

16. En las bases de datos particionadas, existe el archivo db2nodes.cfg y contiene definiciones de particiones (o nodos). Para obtener detalles consulte el manual *Administration Guide*.

17. CURRENT FUNCTION PATH es sinónimo de CURRENT PATH.

El registro especial CURRENT PATH contiene una lista con uno o varios nombres de esquemas, donde éstos aparecen delimitados entre comillas y separados por comas (las comillas dentro de la serie se repiten como se haría con cualquier identificador delimitado).

Por ejemplo, una vía de acceso de SQL que especifica que el gestor de bases de datos primero debe mirar en el esquema FERMAT, luego en XGRAPHIC y por último en SYSIBM se devuelve en el registro especial CURRENT PATH de la siguiente manera:

```
"FERMAT", "XGRAPHIC", "SYSIBM"
```

El valor por omisión es "SYSIBM", "SYSFUN", X, donde X es el valor del registro especial USER delimitado por comillas dobles.

La sentencia SET CURRENT FUNCTION PATH puede cambiar su valor (consulte el apartado "SET PATH" en la página 1181). No es necesario especificar el esquema SYSIBM. Si no se incluye en la vía de acceso de SQL, se supone implícitamente que es el primer esquema. SYSIBM no toma ninguno de los 254 caracteres, si se asume implícitamente.

La utilización de la vía de acceso de SQL para la resolución de funciones se describe en el apartado "Funciones" en la página 159. Un tipo de datos que no está calificado con un nombre de esquema se calificará implícitamente con el primer nombre de esquema que aparezca en la vía de acceso de SQL y que contenga un tipo de datos con el mismo nombre no calificado especificado. Esta regla tiene excepciones, tal y como se describe en las sentencias siguientes: CREATE DISTINCT TYPE, CREATE FUNCTION, COMMENT ON y DROP.

Ejemplo

Utilizando la vista de catálogo SYSCAT.VIEWS, encuentre todas las vistas que se hayan creado exactamente con el mismo valor que el que tiene actualmente el registro especial CURRENT PATH.

```
SELECT VIEWNAME, VIEWSCHEMA FROM SYSCAT.VIEWS
WHERE FUNC_PATH = CURRENT PATH
```

CURRENT QUERY OPTIMIZATION

El registro especial CURRENT QUERY OPTIMIZATION especifica un valor INTEGER que controla la clase de optimización de consulta que realiza el gestor de bases de datos al enlazar sentencias de SQL dinámico. La opción de enlace lógico QUERYOPT controla la clase de optimización de consulta para las sentencias de SQL estáticas (consulte el manual *Consulta de mandatos* para obtener información adicional sobre la opción de enlace QUERYOPT). Los posibles valores oscilan entre 0 y 9. Por ejemplo, si la clase de optimización de consulta se establece en la clase mínima de optimización (0), entonces el valor

Registros especiales

del registro especial es 0. El valor por omisión está determinado por el parámetro de configuración de la base de datos `dft_queryopt`.

La sentencia `SET CURRENT QUERY OPTIMIZATION` puede cambiar su valor (consulte el apartado “`SET CURRENT QUERY OPTIMIZATION`” en la página 1159).

Ejemplo

Utilizando la vista de catálogo `SYSCAT.PACKAGES`, busque todos los planes que se han enlazado con el mismo valor que el valor actual del registro especial `CURRENT QUERY OPTIMIZATION`.

```
SELECT PKGNAME, PKGSHEMA FROM SYSCAT.PACKAGES
WHERE QUERYOPT = CURRENT QUERY OPTIMIZATION
```

CURRENT REFRESH AGE

El registro especial `CURRENT REFRESH AGE` especifica un valor de duración de indicación de la hora con un tipo de datos de `DECIMAL(20,6)`. Esta duración es la duración máxima desde que se ha procesado una sentencia `REFRESH TABLE` de una tabla de resumen `REFRESH DEFERRED` para que la tabla de resumen pueda utilizarse para optimizar el proceso de una consulta. Si `CURRENT REFRESH AGE` tiene un valor de 99 999 999 999 999 (`ANY`) y la clase `QUERY OPTIMIZATION` es 5 o más, se considera que las tablas de resumen `REFRESH DEFERRED` optimizan el proceso de una consulta SQL dinámica. Se supone que una tabla de resumen con el atributo `REFRESH IMMEDIATE` y sin estar en estado pendiente de comprobación tiene un período de renovación de cero.

El valor puede cambiarse mediante la sentencia `SET CURRENT REFRESH AGE` (consulte el apartado “`SET CURRENT REFRESH AGE`” en la página 1162). Las consultas del SQL incorporado estático nunca tienen en cuenta las tablas de resumen definidas con `REFRESH DEFERRED`.

El valor inicial de `CURRENT REFRESH AGE` es cero.

CURRENT SCHEMA

El registro especial `CURRENT SCHEMA` especifica un valor `VARCHAR(128)` que identifica el nombre de esquema utilizado para calificar las referencias a objetos de base de datos no calificadas cuando sea aplicable en sentencias de SQL preparadas dinámicamente.¹⁸

El valor inicial de `CURRENT SCHEMA` es el ID de autorización del usuario de la sesión actual.

18. Para que exista compatibilidad con DB2 para OS/390, el registro especial `CURRENT SQLID` se trata como sinónimo de `CURRENT SCHEMA`.

El valor puede cambiarse mediante la sentencia SET SCHEMA (consulte el apartado “SET SCHEMA” en la página 1184).

La opción de enlace lógico QUALIFIER controla el nombre de esquema utilizado para calificar las referencias de objetos de base de datos no calificadas donde corresponda en las sentencias de SQL estáticas (consulte el manual *Consulta de mandatos* para obtener más información).

Ejemplo

Establezca el esquema para la calificación de objetos en 'D123'.

```
SET CURRENT SCHEMA = 'D123'
```

CURRENT SERVER

El registro especial CURRENT SERVER especifica un valor VARCHAR(18) que identifica el servidor de aplicaciones actual. El nombre real del servidor de aplicaciones (no un seudónimo) está contenido en el registro.

La sentencia CONNECT puede cambiar CURRENT SERVER solamente bajo determinadas circunstancias (consulte el apartado “CONNECT (Tipo 1)” en la página 653).

Ejemplo

Establezca la variable del lenguaje principal APPL_SERVE (VARCHAR(18)) en el nombre del servidor de aplicaciones al que está conectada la aplicación.

```
VALUES CURRENT SERVER  
INTO :APPL_SERVE
```

CURRENT TIME

El registro especial CURRENT TIME especifica una hora basada en la lectura de un reloj cuando la sentencia de SQL se ejecuta en el servidor de aplicaciones. Si este registro especial se utiliza más de una vez en la misma sentencia de SQL o bien con CURRENT DATE o CURRENT TIMESTAMP en una sola sentencia, todos los valores se basan en la misma lectura del reloj.

En un sistema federado, CURRENT TIME se puede utilizar en una consulta destinada a fuentes de datos. Cuando se procesa la consulta, la hora devuelta se obtendrá del registro CURRENT TIME del servidor federado, no de las fuentes de datos.

Ejemplo

Utilizando la tabla CL_SCHED, seleccione todas las clases (CLASS_CODE) que empiezan (STARTING) más tarde hoy. Las clases de hoy tienen un valor 3 en la columna DAY.

```
SELECT CLASS_CODE FROM CL_SCHED  
WHERE STARTING > CURRENT TIME AND DAY = 3
```

CURRENT TIMESTAMP

El registro especial CURRENT TIMESTAMP especifica una indicación de la hora basada en la lectura de un reloj cuando la sentencia de SQL se ejecuta en el servidor de aplicaciones. Si este registro especial se utiliza más de una vez en la misma sentencia de SQL o bien con CURRENT DATE o CURRENT TIME en una sola sentencia, todos los valores se basan en la misma lectura del reloj.

En un sistema federado, CURRENT TIMESTAMP se puede utilizar en una consulta destinada a fuentes de datos. Cuando se procesa la consulta, la indicación de fecha y hora se obtendrá del registro CURRENT TIMESTAMP del servidor federado, no de las fuentes de datos.

Ejemplo

Inserte una fila en la tabla IN_TRAY. El valor de la columna RECEIVED mostrará la indicación de la hora en la que se ha añadido la fila. Los valores de las otras tres columnas se obtienen de las variables del lenguaje principal SRC (char(8)), SUB (char(64)) y TXT (VARCHAR(200)).

```
INSERT INTO IN_TRAY
VALUES (CURRENT_TIMESTAMP, :SRC, :SUB, :TXT)
```

CURRENT TIMEZONE

El registro especial CURRENT TIMEZONE especifica la diferencia entre UTC¹⁹ y la hora local en el servidor de aplicaciones. La diferencia se representa por un período de tiempo (un número decimal cuyos dos primeros dígitos corresponden a las horas, los dos siguientes a los minutos y los dos últimos a los segundos). El número de horas está entre -24 y 24, exclusive. Al restar CURRENT TIMEZONE de la hora local se convierte esa hora a UTC. La hora se calcula a partir de la hora del sistema operativo en el momento en que se ejecuta la sentencia de SQL.²⁰

El registro especial CURRENT TIMEZONE se puede utilizar allí donde se utilice una expresión de tipo de datos DECIMAL(6,0), por ejemplo, en la aritmética de la hora y la indicación de la hora.

Ejemplo

Inserte un registro en la tabla IN_TRAY utilizando una indicación de la hora UTC para la columna RECEIVED.

19. Hora coordinada universal, conocida anteriormente como GMT.

20. El valor de CURRENT TIMEZONE se determina a partir de las funciones de tiempo de ejecución de C. Si desea saber cuáles son los requisitos de instalación relativos a la zona horaria, consulte el manual *Guía rápida de iniciación*.

```
INSERT INTO IN_TRAY VALUES (
    CURRENT_TIMESTAMP - CURRENT_TIMEZONE,
    :source,
    :subject,
    :notetext )
```

USER

El registro especial USER especifica el ID de autorización de tiempo de ejecución que se pasa al gestor de bases de datos cuando se inicia una aplicación en una base de datos. El tipo de datos del registro es VARCHAR(128).

Ejemplo

Seleccione todas las notas de la tabla IN_TRAY colocadas por el propio usuario.

```
SELECT * FROM IN_TRAY
WHERE SOURCE = USER
```

Nombres de columna

El significado de un *nombre de columna* depende de su contexto. Un nombre de columna sirve para:

- Declarar el nombre de una columna como, por ejemplo, en una sentencia CREATE TABLE.
- Identificar una columna como, por ejemplo, en una sentencia CREATE INDEX.
- Especificar los valores de la columna como, por ejemplo, en los contextos siguientes:
 - En una función de columna, un nombre de columna especifica todos los valores de la columna en la tabla de resultado intermedia o de grupo a los que se aplica la función. (Las tablas de resultado intermedias y de grupo se explican en el “Capítulo 5. Consultas” en la página 471.) Por ejemplo, MAX(SALARY) aplica la función MAX a todos los valores de la columna SALARY de un grupo.
 - En una cláusula GROUP BY o ORDER BY, un nombre de columna especifica todos los valores de la tabla de resultado intermedia a los que se aplica la cláusula. Por ejemplo, ORDER BY DEPT ordena una tabla de resultado intermedia según los valores de la columna DEPT.
 - En una expresión, una condición de búsqueda o una función escalar, un nombre de columna especifica un valor para cada fila o grupo al que se aplica la construcción. Por ejemplo, cuando la condición de búsqueda CODE = 20 se aplica a alguna fila, el valor especificado por el nombre de columna CODE es el valor de la columna CODE en esa fila.
- Redenominar temporalmente una columna, como en la *cláusula-correlación* de una *referencia-tabla* en una cláusula FROM.

Nombres de columna

Nombres de columna calificados

Un calificador para un nombre de columna puede ser un nombre de tabla, vista, apodo o seudónimo o un nombre de correlación.

El hecho de que un nombre de columna pueda calificarse depende del contexto:

- Según la forma de la sentencia COMMENT ON, puede que se deba calificar un nombre de una sola columna. No se deben calificar nombres de varias columnas.
- Donde el nombre de columna especifique valores de la columna, puede calificarse como opción del usuario.
- En la cláusula de asignación de una sentencia UPDATE, puede calificarse en la opción del usuario.
- En todos los demás contextos, un nombre de columna no debe calificarse.

Cuando un calificador es opcional, puede cumplir dos finalidades. Estos casos se describen en el apartado “Calificadores de nombres de columna para evitar ambigüedad” en la página 147 y “Calificadores de nombres de columna en referencias correlacionadas” en la página 149.

Nombres de correlación

Un *nombre de correlación* puede definirse en la cláusula FROM de una consulta y en la primera cláusula de una sentencia UPDATE o DELETE. Por ejemplo, la cláusula FROM X.MYTABLE Z establece Z como nombre de correlación para X.MYTABLE.

```
FROM X.MYTABLE Z
```

Con Z definida como nombre de correlación para X.MYTABLE, sólo puede utilizarse Z para calificar una referencia a una columna de esa instancia de X.MYTABLE en esa sentencia SELECT.

Un nombre de correlación sólo se asocia a una tabla, vista, apodo, seudónimo, expresión de tabla anidada o función de tabla sólo dentro del contexto en el que está definido. Por lo tanto, puede definirse el mismo nombre de correlación con distintos propósitos en diferentes sentencias o bien en distintas cláusulas de la misma sentencia.

Como calificador, un nombre de correlación puede utilizarse para evitar ambigüedades o para establecer una referencia correlacionada. También puede utilizarse simplemente como nombre abreviado de una tabla, vista, apodo o seudónimo. En el caso de una expresión de tabla anidada o una función de tabla, es necesario un nombre de correlación para identificar la tabla resultante. En el ejemplo, Z podría haberse utilizado simplemente para evitar tener que entrar X.MYTABLE más de una vez.

Si se especifica un nombre de correlación para una tabla, vista, apodo o seudónimo, cualquier referencia a una columna de esa instancia de la tabla, vista, apodo o seudónimo debe utilizar el nombre de correlación en lugar del nombre de tabla, vista, apodo o seudónimo. Por ejemplo, la referencia a EMPLOYEE.PROJECT del ejemplo siguiente no es correcto, porque se ha especificado un nombre de correlación para EMPLOYEE:

Ejemplo

```
FROM EMPLOYEE E
WHERE EMPLOYEE.PROJECT='ABC'      * incorrecto*
```

La referencia calificada para PROJECT debe utilizar, en su lugar, el nombre de correlación "E", tal como se muestra abajo:

```
FROM EMPLOYEE E
WHERE E.PROJECT='ABC'
```

Los nombres especificados en una cláusula FROM pueden estar *expuestos* o *no expuestos*. Se dice que un nombre de tabla, vista, apodo o seudónimo está expuesto en la cláusula FROM si no se especifica un nombre de correlación. El nombre de la correlación es siempre un nombre expuesto. Por ejemplo, en la siguiente cláusula FROM, se especifica un nombre de correlación para EMPLOYEE pero no para DEPARTMENT, de modo que DEPARTMENT es un nombre expuesto y EMPLOYEE no lo es:

```
FROM EMPLOYEE E, DEPARTMENT
```

Un nombre de tabla, vista, apodo o seudónimo que está expuesto en una cláusula FROM puede ser igual que otro nombre de tabla, vista o apodo expuesto en esa cláusula FROM o cualquier nombre de correlación de la cláusula FROM. Esta situación puede dar como resultado una serie de referencias ambiguas de nombres de columna que acaban devolviendo un código de error (SQLSTATE 42702).

Las dos primeras cláusulas FROM mostradas más abajo son correctas, porque cada una no contiene más de una referencia a EMPLOYEE que esté expuesta:

1. Dada una cláusula FROM:

```
FROM EMPLOYEE E1, EMPLOYEE
```

una referencia calificada como, por ejemplo, EMPLOYEE.PROJECT indica una columna de la segunda instancia de EMPLOYEE en la cláusula FROM. La referencia calificada a la primera instancia de EMPLOYEE debe utilizar el nombre de correlación "E1" (E1.PROJECT).

2. Dada una cláusula FROM:

```
FROM EMPLOYEE, EMPLOYEE E2
```

Nombres de columna

una referencia calificada como, por ejemplo, EMPLEADO.PROYECTO indica una columna de la primera instancia de EMPLEADO en la cláusula FROM. Una referencia calificada a la segunda instancia de EMPLEADO debe utilizar el nombre de correlación "E2" (E2.PROYECTO).

3. Dada una cláusula FROM:

```
FROM EMPLOYEE, EMPLOYEE
```

los dos nombres de tabla expuestos que se incluyen en esta cláusula (EMPLOYEE y EMPLOYEE) son los mismos. Esto está permitido, pero las referencias a nombres de columnas específicos resultarían ambiguas (SQLSTATE 42702).

4. Dada la sentencia siguiente:

```
SELECT *  
FROM EMPLOYEE E1, EMPLOYEE E2           * incorrecto *  
WHERE EMPLOYEE.PROJECT = 'ABC'
```

la referencia calificada EMPLOYEE.PROJECT es incorrecta, porque las dos instancias de EMPLOYEE en la cláusula FROM tienen nombres de correlación. En cambio, las referencias a PROJECT deben estar calificadas con algún nombre de correlación (E1.PROJECT o E2.PROJECT).

5. Dada una cláusula FROM:

```
FROM EMPLOYEE, X.EMPLOYEE
```

una referencia a una columna en la segunda instancia de EMPLOYEE debe utilizar X.EMPLOYEE (X.EMPLOYEE.PROJECT). Si X es el valor del registro especial CURRENT SCHEMA en el SQL dinámico o la opción de precompilación/enlace lógico QUALIFIER en el SQL estático, no se puede hacer ninguna referencia a las columnas porque resultaría ambigua.

La utilización del nombre de correlación en la cláusula FROM permite, también, la opción de especificar una lista de nombres de columna que se han de asociar con las columnas de la tabla resultante. Igual que los nombres de correlación, estos nombres de columna listados se convierten en los nombres *expuestos* de las columnas que deben utilizarse en las referencias a las columnas en toda la consulta. Si se especifica una lista de nombres de columna, los nombres de columna de la tabla principal se convierten en *no expuestos*.

- Dada una cláusula FROM:

```
FROM DEPARTMENT D (NUM,NAME,MGR,ANUM,LOC)
```

una referencia calificada como, por ejemplo, D.NUM indica la primera columna de la tabla DEPARTMENT que se ha definido en la tabla como

DEPTNO. Una referencia a D.DEPTNO utilizando esta cláusula FROM es incorrecta ya que el nombre de columna DEPTNO es un nombre de columna no expuesto.

Calificadores de nombres de columna para evitar ambigüedad

En el contexto de una función, de una cláusula GROUP BY, de una cláusula ORDER BY, de una expresión o de una condición de búsqueda, un nombre de columna hace referencia a los valores de una columna en alguna tabla, vista, apodo, expresión de tabla anidada o función de tabla. Las tablas, vistas, apodos, expresiones de tablas anidadas y funciones de tabla donde puede residir la columna se denominan *tablas de objetos* del contexto. Dos o más tablas de objetos pueden contener columnas con el mismo nombre; un nombre de columna se puede calificar para indicar la tabla de la cual procede la columna. Los calificadores de nombres de columna también son útiles en los procedimientos SQL para diferenciar los nombres de columna de los nombres de variables SQL utilizados en sentencias de SQL.

Una expresión de tabla anidada o una función de tabla trata las *referencias-tabla* que la preceden en la cláusula FROM como tablas de objetos. Las *referencias-tabla* que siguen no se tratan como tablas de objetos.

Designadores de tabla

Un calificador que designa una tabla de objeto específica se conoce como *designador de tabla*. La cláusula que identifica las tablas de objetos también establece los designadores de tabla para ellas. Por ejemplo, las tablas de objetos de una expresión en una cláusula SELECT se nombran en la cláusula FROM que la sigue:

```
SELECT CORZ.COLA, OWNY.MYTABLE.COLA
FROM OWNX.MYTABLE CORZ, OWNY.MYTABLE
```

Los designadores en la cláusula FROM se establecen como sigue:

- Un nombre que sigue a una tabla, vista, apodo, seudónimo, expresión de tabla anidada o función de tabla es a la vez un nombre de correlación y un designador de tabla. Así pues, CORZ es un designador de tabla. CORZ sirve para calificar el primer nombre de columna de la lista de selección.
- Una tabla expuesta, un nombre de vista, un apodo o seudónimo es un designador de tabla. Así pues, OWNY.MYTABLE es un designador de tabla. OWNY.MYTABLE sirve para calificar el nombre de la segunda columna de la lista de selección.

Cada designador de tabla debe ser exclusivo en una cláusula FROM determinada para evitar la aparición de referencias ambiguas a columnas.

Nombres de columna

Evitar referencias no definidas o ambiguas

Cuando un nombre de columna hace referencia a valores de una columna, debe existir una sola tabla de objetos que incluya una columna con ese nombre. Las situaciones siguientes se consideran errores:

- Ninguna tabla de objetos contiene una columna con el nombre especificado. La referencia no está definida.
- El nombre de columna está calificado mediante un designador de tabla, pero la tabla designada no incluye una columna con el nombre especificado. De nuevo, la referencia no está definida.
- El nombre no está calificado, y hay más de una tabla de objetos que incluye una columna con ese nombre. La referencia es ambigua.
- El designador de tabla califica al nombre de columna, pero la tabla designada no es única en la cláusula FROM y ambas ocurrencias de la tabla designada incluyen la columna. La referencia es ambigua.
- El nombre de columna de una expresión de tabla anidada que no va precedida por la palabra clave TABLE o en una función de tabla o expresión de tabla anidada que es el operando derecho de una unión externa derecha o una unión externa completa y el nombre de columna no hace referencia a una columna de una *referencia-tabla* de la selección completa de la expresión de tabla anidada. La referencia no está definida.

Evite las referencias ambiguas calificando un nombre de columna con un designador de tabla definido exclusivamente. Si la columna está en varias tablas de objetos con nombres distintos, los nombres de tabla pueden utilizarse como designadores. Las referencias ambiguas también se pueden evitar sin la utilización del designador de tabla dando nombres exclusivos a las columnas de una de las tablas de objetos utilizando la lista de nombres de columna que siguen al nombre de correlación.

Al calificar una columna con la forma de nombre expuesto de tabla de un designador de tabla, se puede utilizar la forma calificada o no calificada del nombre de tabla expuesto. Sin embargo, el calificador y la tabla utilizados deben ser iguales después de calificar completamente el nombre de tabla, vista o apodo y el designador de tabla.

1. Si el ID de autorización de la sentencia es CORPDATA:

```
SELECT CORPDATA.EMPLOYEE.WORKDEPT
FROM EMPLOYEE
```

es una sentencia válida.

2. Si el ID de autorización de la sentencia es REGION:

```
SELECT CORPDATA.EMPLOYEE.WORKDEPT
FROM EMPLOYEE * incorrecto *
```

no es válido, porque EMPLOYEE representa la tabla REGION.EMPLOYEE, pero el calificador para WORKDEPT representa una tabla distinta, CORPDATA.EMPLOYEE.

Calificadores de nombres de columna en referencias correlacionadas

Una *selección completa* es una forma de consulta que puede utilizarse como componente de varias sentencias de SQL. Consulte el “Capítulo 5. Consultas” en la página 471 para obtener más información sobre las selecciones completas. Una selección completa utilizada dentro de una condición de búsqueda de cualquier sentencia se denomina *subconsulta*. Una selección completa utilizada para recuperar un único valor como, por ejemplo, una expresión en una sentencia se denomina una *selección escalar* o *subconsulta escalar*. Una selección completa utilizada en la cláusula FROM de una consulta se llama *expresión de tabla anidada*. Se hace referencia a las subconsultas de las condiciones de búsqueda, subconsultas escalares y expresiones de tabla anidadas como subconsultas en el resto de este tema.

Una subconsulta puede contener subconsultas propias y éstas, a su vez, pueden contener subconsultas. De este modo, una sentencia de SQL puede contener una jerarquía de subconsultas. Los elementos de la jerarquía que contienen subconsultas están en un nivel superior que las subconsultas que contienen.

Cada elemento de la jerarquía contiene uno o más designadores de tabla. Una consulta puede hacer referencia no solamente a las columnas de las tablas identificadas en su mismo nivel dentro de la jerarquía, sino también a las columnas de las tablas anteriormente identificadas en la jerarquía, hasta alcanzar el estrato más elevado. Una referencia a una columna de una tabla identificada en un nivel superior se llama *referencia correlacionada*.

Para la compatibilidad con los estándares existentes del SQL, se permiten nombres de columna calificados y no calificados como referencias correlacionadas. Sin embargo, es aconsejable calificar todas las referencias de columnas utilizadas en subconsultas; de lo contrario, los nombres de columna idénticos pueden conducir a resultados no deseados. Por ejemplo, si se modifica una tabla de una jerarquía de modo que contenga el mismo nombre de columna que la referencia correlacionada y la sentencia se vuelve a preparar, la referencia se aplicará en la tabla modificada.

Cuando se califica un nombre de columna en una subconsulta, se busca en cada nivel de jerarquía, comenzando en la misma subconsulta en la que aparece el nombre de columna calificado y continuando hacia niveles superiores de la jerarquía, hasta que se encuentre un designador de tabla que coincida con el calificador. Una vez encontrado, se verifica que la tabla contenga la columna en cuestión. Si se encuentra la tabla en un nivel superior que el nivel que contiene el nombre de columna, es que éste es una referencia

Nombres de columna

correlacionada para el nivel donde se encontró el designador de tabla. Una expresión de tabla anidada debe ir precedida por la palabra clave TABLE opcional para buscar en la jerarquía superior la selección completa de la expresión de tabla anidada.

Cuando el nombre de columna de una subconsulta no se califica, se busca en las tablas a las que se hace referencia en cada nivel de la jerarquía, empezando en la misma subconsulta en la que aparece el nombre de columna y siguiendo hacia niveles superiores de la jerarquía hasta que se encuentre un nombre de columna que coincida. Si la columna se encuentra en una tabla en un nivel superior al nivel que contiene el nombre de columna, es que éste es una referencia correlacionada para el nivel donde se ha encontrado la tabla que contiene la columna. Si se encuentra el nombre de columna en más de una tabla en un nivel en concreto, la referencia es ambigua y se considera un error.

En cualquier caso, en el siguiente ejemplo T hace referencia al designador de tabla que contiene la columna C. Un nombre de columna, T.C (donde T representa un calificador implícito o explícito), es una referencia correlacionada solamente si se dan estas condiciones:

- T.C se utiliza en una expresión de una subconsulta.
- T no designa una tabla utilizada en la cláusula de la subconsulta.
- T designa una tabla utilizada en un nivel superior de la jerarquía que contiene la subconsulta.

Debido a que una misma tabla, vista o apodo pueden estar identificados en muchos niveles, se recomienda utilizar nombres de correlación exclusivos como designadores de tabla. Si se utiliza T para designar una tabla en más de un nivel (T es el propio nombre de tabla o es un nombre de correlación duplicado), T.C hace referencia al nivel donde se utiliza T que contiene de forma más directa la subconsulta que incluye T.C. Si es necesario un nivel de correlación superior, debe utilizarse un nombre de correlación exclusivo.

La referencia correlacionada T.C identifica un valor de C en una fila o grupo de T a la que se aplican dos condiciones de búsqueda: la condición 1 en la subconsulta, y la condición 2 en algún nivel superior. Si se utiliza la condición 2 en una cláusula WHERE, se evalúa la subconsulta para cada fila a la que se aplica la condición 2. Si se utiliza la condición 2 en una cláusula HAVING, se evalúa la subconsulta para cada grupo al que se aplica la condición 2. (Para otra explicación sobre la evaluación de subconsultas, consulte las descripciones de las cláusulas WHERE y HAVING en el “Capítulo 5. Consultas” en la página 471.)

Por ejemplo, en la sentencia siguiente, la referencia correlacionada X.WORKDEPT (en la última línea) hace referencia al valor de WORKDEPT en

la tabla EMPLOYEE en el nivel de la primera cláusula FROM. (Dicha cláusula establece X como nombre de correlación para EMPLOYEE.) La sentencia lista los empleados que tienen un salario inferior al promedio de su departamento.

```
SELECT EMPNO, LASTNAME, WORKDEPT
FROM EMPLOYEE X
WHERE SALARY < (SELECT AVG(SALARY)
FROM EMPLOYEE
WHERE WORKDEPT = X.WORKDEPT)
```

El ejemplo siguiente utiliza ESTE como nombre de correlación. La sentencia elimina las filas de los departamentos que no tienen empleados.

```
DELETE FROM DEPARTMENT THIS
WHERE NOT EXISTS(SELECT *
FROM EMPLOYEE
WHERE WORKDEPT = ESTE.DEPTNO)
```

Referencias a variables del lenguaje principal

Una *variable del lenguaje principal* es:

- Una variable de un lenguaje principal como por ejemplo, una variable C, una variable C++, un elemento de datos COBOL, una variable FORTRAN o una variable Java

o:

- Una construcción del lenguaje principal generada por un precompilador SQL a partir de una variable declarada mediante extensiones SQL

a la que se hace referencia en una sentencia de SQL. Las variables de lenguaje principal se definen directamente mediante las sentencias del lenguaje principal o indirectamente mediante extensiones SQL.

Una variable del lenguaje principal en una sentencia de SQL debe identificar una variable del lenguaje principal descrita en el programa según las normas para la declaración de variables del lenguaje principal.

Todas las variables del lenguaje principal utilizadas en una sentencia de SQL deben estar declaradas en una sección DECLARE del SQL en todos los lenguajes principales excepto en REXX (consulte el manual *Application Development Guide* para obtener más información acerca de la declaración de variables del lenguaje principal para sentencias de SQL de programas de aplicación). No se debe declarar ninguna variable fuera de una sección DECLARE del SQL con nombres que sean idénticos a variables declaradas en una sección DECLARE del SQL. Una sección DECLARE del SQL empieza por BEGIN DECLARE SECTION y termina por END DECLARE SECTION.

La metavariante *variable-lengprinc*, tal como se utiliza en los diagramas de sintaxis, muestra una referencia a una variable del lenguaje principal. Una

Referencias a variables del lenguaje principal

variable de lenguaje principal en la cláusula VALUES INTO o en la cláusula INTO de una sentencia FETCH o SELECT INTO, identifica una variable de lenguaje principal a la que se asigna un valor procedente de una columna de una fila o una expresión. En todos los demás contextos, una variable-lengprinc especifica un valor que ha de pasarse al gestor de bases de datos desde el programa de aplicación.

Variables del lenguaje principal en el SQL dinámico

En sentencias de SQL dinámicas, se utilizan los marcadores de parámetros en lugar de las variables del lenguaje principal. Un marcador de parámetros es un signo de interrogación (?) que representa una posición en una sentencia de SQL dinámica en la que la aplicación proporcionará un valor; es decir, donde se encontrará una variable de lenguaje principal si la serie de la sentencia es una sentencia de SQL estática. El siguiente ejemplo muestra una sentencia de SQL estática que emplea variables de lenguaje principal:

```
INSERT INTO DEPARTMENT
VALUES (:hv_deptno, :hv_deptname, :hv_mgrno, :hv_admrdept)
```

Este ejemplo muestra una sentencia de SQL dinámica que utiliza marcadores de parámetros:

```
INSERT INTO DEPARTMENT VALUES (?, ?, ?, ?)
```

Para obtener más información sobre los marcadores de parámetros, consulte el apartado “Marcadores de parámetros” de “PREPARE” en la página 1099.

Generalmente, la metavARIABLE *variable-lengprinc* se puede expandir en los diagramas de sintaxis a:



Cada *identificador-lengprinc* debe declararse en el programa fuente. La variable designada por el segundo *identificador-lengprinc* debe tener un tipo de datos de entero pequeño.

El primer *identificador-lengprinc* designa la *variable principal*. Según la operación, proporciona un valor al gestor de bases de datos o bien el gestor de bases de datos le proporciona un valor. Una variable del lenguaje principal de entrada proporciona un valor en la página de códigos de la aplicación en tiempo de ejecución. A la variable del lenguaje principal de salida se le proporciona un valor que, si es necesario, se convierte a la página de códigos de la aplicación en tiempo de ejecución cuando los datos se copian en la

Referencias a variables del lenguaje principal

variable de la aplicación de salida. Una variable del lenguaje principal determinada puede servir tanto de variable de entrada como de salida en el mismo programa.

El segundo identificador-lengprinc designa su *variable indicadora*. La finalidad de la variable indicadora es:

- Especificar el valor nulo. Un valor negativo de la variable indicadora especifica el valor nulo. Un valor de -2 indica una conversión numérica o un error de expresión aritmética ocurrido al obtener el resultado
- Registra la longitud original de una serie truncada (si la fuente del valor no es un tipo de gran objeto)
- Registra la parte correspondiente a los segundos de una hora si la hora se trunca al asignarse a una variable del lenguaje principal.

Por ejemplo, si se utiliza :HV1:HV2 para especificar un valor de inserción o de actualización y si HV2 es negativo, el valor especificado es el valor nulo. Si HV2 no es negativo, el valor especificado es el valor de HV1.

Del mismo modo, si se especifica :HV1:HV2 en una cláusula VALUES INTO o en una sentencia FETCH o SELECT INTO y si el valor devuelto es nulo, HV1 no se cambia y HV2 se establece en un valor negativo.²¹ Si el valor devuelto no es nulo, se asigna dicho valor a HV1 y HV2 se establece en cero (a no ser que la asignación a HV1 necesite el truncamiento de una serie que sea no LOB, en cuyo caso HV2 se establece en la longitud original de la serie). Si una asignación necesita el truncamiento de la parte correspondiente a los segundos de una hora, HV2 se establece en el número de segundos.

Si se omite el segundo identificador del lenguaje principal, la variable de lenguaje principal carece de variable indicadora. El valor especificado por la referencia a la variable del lenguaje principal :HV1 siempre es el valor de HV1 y los valores nulos no se pueden asignar a la variable. Por este motivo, esta forma no debe utilizarse en una cláusula INTO a no ser que la columna correspondiente no pueda incluir valores nulos. Si se utiliza esta forma y la columna contiene valores nulos, el gestor de bases de datos generará un error en tiempo de ejecución.

Una sentencia de SQL que haga referencia a variables de lenguaje principal debe pertenecer al ámbito de la declaración de esas variables de lenguaje

21. Si se configura la base de datos con DFT_SQLMATHWARN en sí (o lo es durante el enlace lógico de una sentencia de SQL estática), HV2 podría ser -2. Si HV2 es -2, no podría devolverse un valor para HV1 debido a un error en la conversión al tipo numérico de HV1 o un error al evaluar una expresión aritmética que se utiliza para determinar el valor para HV1. Cuando se accede a una base de datos con una versión cliente anterior a DB2 Universal Database Versión 5, HV2 será -1 para las excepciones aritméticas.

Referencias a variables del lenguaje principal

principal. En cuanto a las variables a las que la sentencia SELECT del cursor hace referencia, esa regla se aplica más a la sentencia OPEN que a la sentencia DECLARE CURSOR.

Ejemplo

Utilizando la tabla PROJECT, asigne a la variable de lenguaje principal PNAME (VARCHAR(26)) el nombre de proyecto (PROJNAME), a la variable de lenguaje principal STAFF (dec(5,2)) el nivel principal de personal (PRSTAFF) y a la variable de lenguaje principal MAJPROJ (char(6)) el proyecto principal (MAJPROJ) para el proyecto (PROJNO) 'IF1000'. Las columnas PRSTAFF y MAJPROJ pueden contener valores nulos, por lo tanto proporcione las variables indicadoras STAFF_IND (smallint) y MAJPROJ_IND (smallint).

```
SELECT PROJNAME, PRSTAFF, MAJPROJ
INTO :PNAME, :STAFF :STAFF_IND, :MAJPROJ :MAJPROJ_IND
FROM PROJECT
WHERE PROJNO = 'IF1000'
```

Consideraciones acerca de MBCS: El hecho de que los caracteres de múltiples bytes se puedan utilizar en un nombre de variable del lenguaje principal depende del lenguaje principal.

Referencias a las variables del lenguaje principal BLOB, CLOB y DBCLOB

Las variables regulares BLOB, CLOB y DBCLOB, las variables localizadoras LOB (vea “Referencias a variables localizadoras” en la página 155), y las variables de referencia a archivos LOB (vea “Referencias a las variables de referencia a archivos BLOB, CLOB y DBCLOB” en la página 155) se pueden definir en todos los lenguajes principales. Donde se pueden utilizar valores LOB, el término *variable-lengprinc* en un diagrama de sintaxis puede hacer referencia a una variable de lenguaje principal normal, a una variable localizadora o a una variable de referencia a archivos. Puesto que no son tipos de datos nativos, se utilizan las extensiones SQL y los precompiladores generan las construcciones de lenguaje principal necesarias para poder representar a cada variable. En cuanto a REXX, las variables LOB se correlacionan con series.

A veces es posible definir una variable lo suficientemente grande como para contener todo un valor de gran objeto. Si es así y no hay ninguna ventaja de rendimiento si se utiliza la transferencia diferida de datos desde el servidor, no es necesario un localizador. No obstante, puesto que el lenguaje principal o las restricciones de espacio se oponen al almacenamiento de un gran objeto entero en el almacenamiento temporal de una vez o por motivos de rendimiento, se puede hacer referencia a un gran objeto por medio de un localizador y las partes de dicho objeto se pueden seleccionar o actualizar en las variables de lenguaje principal que contengan sólo una parte del gran objeto.

Referencias a variables del lenguaje principal

Como sucede con el resto de variables de lenguaje principal, una variable localizadora de LOB puede tener asociada una variable indicadora. Las variables indicadoras para las variables de lenguaje principal de localizador de gran objeto funcionan de la misma manera que las variables de indicador de otros tipos de datos. Cuando una base de datos devuelve un valor nulo, se define la variable indicadora y la variable localizadora de lenguaje principal no se cambia. Esto significa que un localizador jamás puede apuntar a un valor nulo.

Referencias a variables localizadoras

Una *variable localizadora* es una variable del lenguaje principal que contiene el localizador que representa a un valor LOB en el servidor de aplicaciones. (Vea “Manipulación de grandes objetos (LOB) con localizadores” en la página 85 para obtener información sobre la forma en que se pueden utilizar los localizadores para manipular valores LOB.)

Una variable localizadora de una sentencia de SQL debe identificar una variable localizadora descrita en el programa de acuerdo a las reglas de declaración de variables localizadoras. Siempre se produce indirectamente a través de una sentencia de SQL.

El término variable localizadora, tal como se utiliza en los diagramas de sintaxis, muestra una referencia a una variable localizadora. La metavariante *variable-localizadora* puede expandirse para que incluya un *identificador-lengprinc* igual que para la *variable-lengprinc*.

Cuando la variable de indicador asociada con un localizador es nula, el valor del LOB al que se hace referencia también es nulo.

Si se hace referencia a una variable localizadora que en ese momento no represente ningún valor, se producirá un error (SQLSTATE 0F001).

Durante la confirmación de la transacción, o en cualquier finalización de transacción, se liberan todos los localizadores que la transacción había adquirido.

Referencias a las variables de referencia a archivos BLOB, CLOB y DBCLOB

Las variables de referencia a archivos BLOB, CLOB y DBCLOB sirven para la entrada y salida directa de archivo para los LOB y pueden definirse en todos los lenguajes principales. Puesto que no son tipos de datos nativos, se utilizan las extensiones SQL y los precompiladores generan las construcciones de lenguaje principal necesarias para poder representar a cada variable. En cuanto a REXX, las variables LOB se correlacionan con series.

Referencias a variables del lenguaje principal

Una variable de referencia a archivos representa (más que contiene) al archivo, de igual manera que un localizador de LOB representa, más que contiene, a los bytes LOB. Las consultas, actualizaciones e inserciones pueden utilizar variables de referencia a archivos para almacenar o recuperar valores de una sola columna.

Una variable de referencia a archivos tiene las siguientes propiedades:

Tipo de datos	BLOB, CLOB o DBCLOB. Esta propiedad se especifica al declarar la variable.
Dirección	<p>La dirección debe ser especificada por el programa de aplicación durante la ejecución (como parte del valor de Opciones de archivo). La dirección puede ser:</p> <ul style="list-style-type: none">• De entrada (se utiliza como fuente de datos en las sentencias EXECUTE, OPEN, UPDATE, INSERT o DELETE).• De salida (se utiliza como datos de destino en sentencias las FETCH o SELECT INTO).
Nombre del archivo	<p>Debe especificarlo el programa de aplicación en tiempo de ejecución. Puede ser:</p> <ul style="list-style-type: none">• El nombre completo de la vía de acceso de un archivo (opción que se recomienda).• Un nombre de archivo relativo. Si se proporciona un nombre de archivo relativo, se añade a la vía de acceso actual del proceso cliente. <p>En una aplicación, sólo debe hacerse referencia a un archivo en una variable de referencia a archivos.</p>
Longitud del nombre de archivo	Debe especificarlo el programa de aplicación en tiempo de ejecución. Es la longitud del nombre de archivo (en bytes).
Opciones de archivo	Una aplicación debe asignar una las opciones a una variable de referencia a archivos antes de utilizar dicha variable. Las opciones se establecen mediante un valor INTEGER en un campo de la estructura de la variable de referencia a archivos. Se debe especificar alguna de estas opciones para cada variable de referencia a archivos:

Referencias a variables del lenguaje principal

- Entrada (de cliente a servidor)

SQL_FILE_READ²²

Archivo regular que se puede abrir, leer y cerrar.

- Salida (de servidor a cliente)

SQL_FILE_CREATE²³

Crear un nuevo archivo. Si el archivo ya existe, se produce un error.

SQL_FILE_OVERWRITE (Overwrite)²⁴

Si ya existe un archivo con el nombre especificado, se sobrescribe el contenido del archivo; de lo contrario, se crea un nuevo archivo.

SQL_FILE_APPEND²⁵

Si ya existe un archivo con el nombre especificado, la salida se añade a éste; de lo contrario, se crea un nuevo archivo.

Longitud de datos

No se utiliza en la entrada. En la salida, la implantación establece la longitud de datos en la longitud de los nuevos datos grabados en el archivo. La longitud se mide en bytes.

Como sucede con el resto de variables del lenguaje principal, una variable de referencia a archivos puede tener asociada una variable de indicador.

Ejemplo de una variable de referencia a archivos de salida (en C)

- Suponga una sección de declaración codificada como:

```
EXEC SQL BEGIN DECLARE SECTION
      SQL TYPE IS CLOB_FILE hv_text_file;
      char hv_patent_title[64];
EXEC SQL END DECLARE SECTION
```

22. SQL-FILE-READ en COBOL, sql_file_read en FORTRAN, READ en REXX.

23. SQL-FILE-CREATE en COBOL, sql_file_create en FORTRAN, CREATE en REXX.

24. SQL-FILE-OVERWRITE en COBOL, sql_file_overwrite en FORTRAN, OVERWRITE en REXX.

25. SQL-FILE-APPEND en COBOL, sql_file_append en FORTRAN, APPEND en REXX.

Referencias a variables del lenguaje principal

Una vez procesada:

```
EXEC SQL BEGIN DECLARE SECTION
/* SQL TYPE IS CLOB_FILE hv_text_file; */
struct {
    unsigned long name_length; // Longitud del nombre del archivo
    unsigned long data_length; // Longitud de datos
    unsigned long file_options; // Opciones de archivo
    char name[255]; // Nombre del archivo
} hv_text_file;
char hv_patent_title[64];
EXEC SQL END DECLARE SECTION
```

El código siguiente puede utilizarse para seleccionar en una columna CLOB de la base de datos para un nuevo archivo al que :hv_text_file hace referencia.

```
strcpy(hv_text_file.name, "/u/gainer/papers/sigmod.94");
hv_text_file.name_length = strlen("/u/gainer/papers/sigmod.94");
hv_text_file.file_options = SQL_FILE_CREATE;

EXEC SQL SELECT content INTO :hv_text_file from papers
WHERE TITLE = 'The Relational Theory behind Juggling';
```

Ejemplo de una variable de referencia a archivos de entrada (en C)

- Tomando la misma sección de declaración que antes, se puede utilizar el siguiente código para insertar datos de un archivo normal al que :hv_text_file hace referencia en una columna CLOB.

```
strcpy(hv_text_file.name, "/u/gainer/patents/chips.13");
hv_text_file.name_length = strlen("/u/gainer/patents/chips.13");
hv_text_file.file_options = SQL_FILE_READ;
strcpy(:hv_patent_title, "A Method for Pipelining Chip Consumption");

EXEC SQL INSERT INTO patents( title, text )
VALUES(:hv_patent_title, :hv_text_file);
```

Referencias a variables de lenguaje principal de tipo estructurado

Las variables de tipo estructurado se pueden definir en todos los lenguajes principales, excepto FORTRAN, REXX y Java. Puesto que no son tipos de datos nativos, se utilizan las extensiones SQL y los precompiladores generan las construcciones de lenguaje principal necesarias para poder representar a cada variable.

Al igual que en todas las demás variables de lenguaje principal, una variable de tipo estructurado puede tener una variable indicadora asociada. Las variables indicadoras correspondientes a las variables de lenguaje principal de tipo estructurado actúan de la misma manera que las variables indicadoras de otros tipos de datos. Cuando una base de datos devuelve un valor nulo, se define la variable indicadora y la variable de lenguaje principal de tipo estructurado no cambia.

Referencias a variables del lenguaje principal

La variable de lenguaje principal propiamente dicha correspondiente a un tipo estructurado está definida como tipo de datos interno. El tipo de datos interno asociado al tipo estructurado debe ser asignable:

- desde el resultado de la función de transformación FROM SQL para el tipo estructurado tal como está definida por la opción especificada TRANSFORM GROUP del mandato de precompilación; y
- al parámetro de la función de transformación TO SQL para el tipo estructurado tal como está definida por la opción especificada TRANSFORM GROUP del mandato de precompilación.

Si se utiliza un marcador de parámetros en lugar de una variable de lenguaje principal, se deben especificar las características apropiadas del tipo de parámetro en la SQLDA. Esto requiere un conjunto "duplicado" de estructuras SQLVAR en la SQLDA, y el campo SQLDATATYPE_NAME de la SQLVAR debe contener el nombre de esquema y nombre de tipo del tipo estructurado. Si se omite el esquema en la estructura SQLDA, se produce un error (SQLSTATE 07002). Para obtener más información sobre este tema, vea el "Apéndice C. Área de descriptores SQL (SQLDA)" en la página 1267.

Ejemplo

Defina las variables de lenguaje principal *hv_poly* y *hv_point* (de tipo POLYGON, utilizando el tipo interno BLOB(1048576)) en un programa C.

```
EXEC SQL BEGIN DECLARE SECTION;  
SQL estático  
        TYPE IS POLYGON AS BLOB(1M)  
        hv_poly, hv_point;  
EXEC SQL END DECLARE SECTION;
```

Funciones

Una *función de base de datos* es una relación entre un conjunto de valores de datos de entrada y un conjunto de valores del resultado. Por ejemplo, se pueden pasar a la función `TIMESTAMP` valores de datos de entrada del tipo `DATE` y `TIME`; el resultado será `TIMESTAMP`. Las funciones pueden ser incorporadas o bien definidas por el usuario.

- Las *funciones incorporadas* se proporcionan con el gestor de bases de datos proporcionando un solo valor del resultado y se identifican como parte del esquema `SYSIBM`. Ejemplos de dichas funciones son las funciones de columna como `AVG`, funciones de operador como `"+"`, funciones de conversión como `DECIMAL` y otras como `SUBSTR`.
- Las *funciones definidas por el usuario* son funciones que están registradas en una base de datos de `SYSCAT.FUNCTIONS` (utilizando la sentencia `CREATE FUNCTION`). Estas funciones nunca forman parte del esquema `SYSIBM`. El gestor de bases de datos proporciona un conjunto de estas funciones dentro de un esquema denominado `SYSFUN`.

Funciones

Con las funciones definidas por el usuario, DB2 permite a los usuarios y a los desarrolladores de aplicaciones ampliar la función del sistema de bases de datos añadiendo definiciones de funciones proporcionadas por los usuarios u otros proveedores para aplicar en la máquina de la base de datos en sí. Esto permite un mayor rendimiento que la recuperación de filas de la base de datos y la aplicación de las funciones en los datos recuperados para calificar con más profundidad o para realizar una reducción de datos. La ampliación de las funciones de la base de datos también permite que la base de datos explote las mismas funciones en la máquina que las que utiliza una aplicación, proporciona más sinergia entre la aplicación y la base de datos y contribuye a una mayor productividad para los desarrolladores de aplicaciones porque está más orientada a objetos.

Encontrará una lista completa de funciones de los esquemas SYSIBM y SYSFUN en la Tabla 15 en la página 236.

Funciones definidas por el usuario externas, de SQL y derivadas

Una función definida por el usuario puede ser una *función externa*, una *función de SQL* o una *función derivada*. Una *función externa* se define en la base de datos con una referencia a una biblioteca de códigos objeto y una función en dicha biblioteca que se ejecutará cuando se invoque la función. Las funciones externas no pueden ser funciones de columna. Una *función derivada* se define para la base de datos con una referencia a otra función incorporada o definida por el usuario que ya se conoce en la base de datos. Las funciones derivadas pueden ser funciones escalares o funciones de columna. Son muy útiles para dar soporte a la utilización de funciones existentes con tipos definidos por el usuario. Una *función de SQL* se define para la base de datos utilizando solamente la sentencia RETURN de SQL. Puede devolver un valor escalar, una fila o una tabla. Las funciones de SQL no pueden ser funciones de columna.

Funciones definidas por el usuario: escalares, de columna, de fila y de tabla

Las funciones definidas por el usuario pueden también clasificarse como funciones *escalares*, de *columna* y de *tabla*.

Una *función escalar* es la que devuelve una respuesta de un solo valor cada vez que se invoca. Por ejemplo, la función incorporada SUBSTR() es una función escalar. Las UDF escalares pueden ser externas o derivadas.

Una *función de columna* es la que recibe un conjunto de valores similares (una columna) y devuelve una respuesta de un solo valor. A veces éstas se denominan también *funciones de agregación* en DB2. Un ejemplo de una función de columna es la función incorporada AVG(). No puede definirse UDF de columna externa para DB2, pero puede definirse una UDF de columna derivada de las funciones de columna incorporadas. Es útil para tipos diferenciados. Por ejemplo si está definido un tipo diferenciado

SHOESIZE con un tipo base INTEGER, se podría definir una UDF AVG(SHOESIZE) que tuviese su fuente en la función incorporada AVG(INTEGER) y sería una función de columna.

Una *función de fila* es una función que devuelve una fila de valores. Se puede utilizar sólo como función de transformación, que correlaciona valores de atributos de un tipo estructurado con valores de una fila. Las funciones de fila deben estar definidas como funciones de SQL.

Una *función de tabla* es una función que devuelve una tabla a la sentencia de SQL donde se invoca la función. Sólo se puede invocar la función en la cláusula FROM de una sentencia SELECT. Una función así puede utilizarse para aplicar la potencia de proceso del lenguaje SQL a datos que no son de DB2, o para convertir tales datos a una tabla DB2. Esta función podría, por ejemplo, tomar un archivo y convertirlo en una tabla, tomar muestras de datos de la Web y disponerlos en forma de tabla o acceder a una base de datos Lotus Notes y devolver información sobre mensajes de correo, tal como la fecha, el remitente y el texto del mensaje. Esta información puede unirse a otras tablas de la base de datos. Una función de tabla se puede definir como función externa o como función de SQL (una función de tabla no puede ser una función derivada).

Signaturas de función

Una función se identifica por su esquema, un nombre de función el número de parámetros y los tipos de datos de sus parámetros. Esto se denomina *signatura de función*, que debe ser exclusiva en la base de datos. Varias funciones pueden tener el mismo nombre dentro de un esquema, siempre que el número de parámetros o bien los tipos de datos de los parámetros sean diferentes. Un nombre de función para el que existen múltiples instancias de función se llama función *sobrecargada*. Un nombre de función se puede sobrecargar dentro de un esquema, lo que significa que hay más de una función (y, por consiguiente, diferentes tipos de parámetros) que tiene ese nombre dentro del esquema. Un nombre de función también puede sobrecargarse en una vía de acceso de SQL, en cuyo caso hay más de una función con ese nombre en la vía de acceso y dichas funciones no tienen necesariamente tipos de parámetros diferentes.

Vía de acceso de SQL

Una función se puede invocar haciendo referencia, dentro de un contexto permitido, al nombre calificado (esquema y nombre de función) seguido por la lista de argumentos, encerrados entre paréntesis. También se puede invocar sin especificar el nombre del esquema obteniendo como resultado una serie de posibles funciones de diferentes esquemas que tienen los mismos parámetros o bien parámetros aceptables. En este caso, la *vía de acceso de SQL* se utiliza como ayuda en la *resolución de función*. La vía de acceso de SQL es una lista de esquemas que se examinan para identificar una función con el mismo nombre,

Funciones

número de parámetros y tipos de datos aceptables. Para las sentencias de SQL estáticas, la vía de acceso de SQL se especifica utilizando la opción de enlace lógico FUNCPATH (para obtener detalles, consulte el manual *Consulta de mandatos*). Para las sentencias de SQL dinámicas, la vía de acceso de SQL es el valor del registro especial CURRENT PATH (vea “CURRENT PATH” en la página 138).

Resolución de funciones

En la invocación de una función determinada, el gestor de bases de datos debe decidir, entre las funciones con el mismo nombre, cual es la “mejor” opción. Esto incluye la resolución de funciones incorporadas y definidas por el usuario.

Un *argumento* es un valor que se pasa a una función en una invocación. Cuando se invoca una función en SQL, se pasa una lista de cero o más argumentos. Son argumentos posicionales en tanto que la semántica de dichos argumentos viene determinada por su posición en la lista de argumentos. Un *parámetro* es una definición formal de una entrada para una función. Cuando se define una función en la base de datos, ya sea internamente (funciones incorporadas) o por el usuario (funciones definidas por el usuario), se especifican sus parámetros (cero o más), el orden de las definiciones que determinan su posición y por lo tanto la semántica. De este modo, cada parámetro es una entrada posicional particular de una función. En la invocación, un argumento corresponde a un parámetro determinado en virtud a la posición que éste ocupe en la lista de argumentos.

El gestor de bases de datos utiliza el nombre de la función que se facilita en la invocación, el número y los tipos de datos de los argumentos, todas las funciones que tienen el mismo nombre en la vía de acceso de SQL y los tipos de datos de sus parámetros correspondientes como punto de referencia para decidir si se selecciona o no una función. A continuación se muestran los resultados posibles del proceso de decisión:

1. Una función determinada se considera como la mejor. Por ejemplo, con las funciones denominadas RISK en el esquema TEST con las firmas definidas como:

```
TEST.RISK(INTEGER)
TEST.RISK(DOUBLE)
```

una vía de acceso de SQL que incluya el esquema TEST y la siguiente referencia de función (donde DB es una columna DOUBLE):

```
SELECT ... RISK(DB) ...
```

se elegirá el segundo RISK.

La siguiente referencia de función (donde SI es una columna SMALLINT):

```
SELECT ... RISK(SI) ...
```

elegirá el primer RISK, puesto que SMALLINT puede promocionarse a INTEGER y constituye una coincidencia mejor que DOUBLE, que se encuentra más abajo en la lista de prioridad (como se muestra en la Tabla 5 en la página 100).

Cuando se tienen en cuenta argumentos que son tipos estructurados, la lista de prioridad incluye los supertipos del tipo estático del argumento. La función que mejor se ajusta es la definida con el parámetro de supertipo más cercano, en la jerarquía de tipos estructurados, al tipo estático del argumento de función.

2. Ninguna función se considera la mejor. Tomando como ejemplo las dos mismas funciones del caso anterior y la siguiente referencia de función (donde C es una columna CHAR(5)):

```
SELECT ... RISK(C) ...
```

el argumento es incoherente con el parámetro de las dos funciones RISK.

3. Una función determinada se selecciona según la vía de acceso de SQL y el número de argumentos, así como sus tipos de datos, que se han pasado en la invocación. Por ejemplo, dadas unas funciones denominadas RANDOM con las signaturas definidas como:

```
TEST.RANDOM(INTEGER)
PROD.RANDOM(INTEGER)
```

y una vía de acceso de SQL de:

```
"TEST", "PROD"
```

la siguiente referencia de función:

```
SELECT ... RANDOM(432) ...
```

elegirá TEST.RANDOM puesto que las dos funciones RANDOM son coincidencias igualmente buenas (exactas en este caso particular) y ambos esquemas están en la vía de acceso, pero TEST precede a PROD en la vía de acceso de SQL.

Método de elección de la mejor opción

La comparación de los tipos de datos de los argumentos con los tipos de datos definidos de los parámetros de las funciones en cuestión, constituye la base primordial para tomar la decisión de las funciones de un grupo de

Funciones

funciones con la misma denominación que se consideren "mejores". Tenga en cuenta que el tipo de datos del resultado de la función o el tipo de función (columna, escalar o tabla) en consideración **no** entra en esta determinación.

La resolución de función se consigue siguiendo estas indicaciones.

1. En primer lugar, detecte todas las funciones del catálogo (SYSCAT.FUNCTIONS) y las funciones incorporadas de modo que todas las condiciones siguientes sean ciertas:
 - a. En las invocaciones donde se ha especificado el nombre de esquema (es decir, una referencia calificada), el nombre de esquema y el nombre de función coincidan con el nombre de invocación.
 - b. En las invocaciones donde no se ha especificado el nombre de esquema (es decir, una referencia no calificada), el nombre de función coincide con el nombre de invocación y tiene un nombre de esquema que coincide con uno de los esquemas de la vía de acceso de SQL.
 - c. El número de parámetros definidos debe coincidir con la invocación.
 - d. Cada argumento de invocación coincide con el parámetro definido correspondiente de la función en el tipo de datos o es "promocionable" a ese tipo (consulte el apartado "Promoción de los tipos de datos" en la página 100).
2. A continuación, examine de izquierda a derecha cada argumento de la invocación de la función. En cada argumento, elimine todas las funciones que no sean la *mejor coincidencia* para dicho argumento. La *mejor coincidencia* para un argumento dado es el primer tipo de datos que aparece en la lista de prioridad correspondiente al tipo de datos del argumento en la Tabla 5 en la página 100 para el que exista una función con un parámetro de dicho tipo de datos. En esta comparación no se tienen en cuenta las longitudes, precisiones y escalas ni el atributo "FOR BIT DATA". Por ejemplo, un argumento DECIMAL(9,1) se considera una coincidencia exacta para un parámetro DECIMAL(6,5) mientras que un argumento VARCHAR(19) es una coincidencia exacta para un parámetro VARCHAR(6).

La mejor coincidencia para un argumento de tipo estructurado definido por el usuario es el propio argumento; la siguiente mejor coincidencia es el supertipo inmediato, y así sucesivamente para cada supertipo del argumento. Observe que sólo se tiene en cuenta el tipo estático (tipo declarado) del argumento de tipo estructurado, no el tipo dinámico (tipo más específico).

3. Si después del Paso 2 queda más de una función elegible, es debido a que (a causa de la forma en que funciona el algoritmo) todas las funciones elegibles restantes tienen firmas idénticas, pero se encuentran en esquemas diferentes. Elija la función cuyo esquema sea el primero en la vía de acceso de SQL del usuario.

4. Si después del Paso 2 no queda ninguna función elegible, se devolverá un error (SQLSTATE 42884).

Consideraciones de vía de acceso de función para funciones incorporadas

Las funciones incorporadas residen en un esquema especial denominado SYSIBM. Hay funciones adicionales disponibles en el esquema SYSFUN que no se consideran funciones incorporadas ya que se han desarrollado como funciones definidas por el usuario y carecen de consideraciones de proceso especiales. Los usuarios no pueden definir funciones adicionales en los esquemas SYSIBM o SYSFUN (ni en cualquier esquema cuyo nombre empiece por las letras "SYS").

Como ya se ha indicado, las funciones incorporadas participan en el proceso de resolución de función exactamente como lo hacen las funciones definidas por el usuario. Una diferencia entre ambas, desde la perspectiva de resolución de funciones, es que la resolución de funciones siempre tiene que tener en cuenta las funciones incorporadas. Por este motivo, la resolución de tipos de datos y de funciones da por sentado que SYSIBM es el primer esquema de la vía de acceso si en ella no aparece el nombre de esquema SYSIBM.

Por ejemplo, si la vía de acceso de SQL del usuario se define de la siguiente manera:

```
"SHAREFUN", "SYSIBM", "SYSFUN"
```

y hay una función LENGTH definida en el esquema SHAREFUN con el mismo número y los mismos tipos de argumentos que SYSIBM.LENGTH, una referencia no calificada a LENGTH en la sentencia de SQL de este usuario hará que se seleccione SHAREFUN.LENGTH. No obstante, si la vía de acceso de SQL del usuario se define de la siguiente forma:

```
"SHAREFUN", "SYSFUN"
```

y existe la misma función SHAREFUN.LENGTH, una referencia no calificada a LENGTH en la sentencia de SQL de este usuario hará que se seleccione SYSIBM.LENGTH, puesto que SYSIBM está implícitamente en primer lugar en la vía de acceso porque no se había especificado.

Se pueden minimizar los posibles problemas en este área si:

- no se utilizan nunca nombres de funciones incorporadas para las funciones definidas por el usuario, ni
- se califica ninguna referencia a estas funciones, si por algún motivo se considera necesario crear una función definida por el usuario con el mismo nombre que una función incorporada.

Ejemplo de resolución de función

A continuación se muestra un ejemplo de una resolución de función satisfactoria.

Existen siete funciones ACT, en tres esquemas diferentes, registradas del modo siguiente (observe que no aparecen todas las palabras clave necesarias):

```
CREATE FUNCTION AUGUSTUS.ACT (CHAR(5), INT, DOUBLE) SPECIFIC ACT_1 ...
CREATE FUNCTION AUGUSTUS.ACT (INT, INT, DOUBLE) SPECIFIC ACT_2 ...
CREATE FUNCTION AUGUSTUS.ACT (INT, INT, DOUBLE, INT) SPECIFIC ACT_3 ...
CREATE FUNCTION JULIUS.ACT (INT, DOUBLE, DOUBLE) SPECIFIC ACT_4 ...
CREATE FUNCTION JULIUS.ACT (INT, INT, DOUBLE) SPECIFIC ACT_5 ...
CREATE FUNCTION JULIUS.ACT (SMALLINT, INT, DOUBLE) SPECIFIC ACT_6 ...
CREATE FUNCTION NERO.ACT (INT, INT, DEC(7,2)) SPECIFIC ACT_7 ...
```

La referencia de función es la siguiente (donde I1 e I2 son columnas INTEGER y D es una columna DECIMAL):

```
SELECT ... ACT(I1, I2, D) ...
```

Suponga que la aplicación que efectúa esta referencia tiene una vía de acceso de SQL establecida como:

```
"JULIUS", "AUGUSTUS", "CAESAR"
```

Como resultado del algoritmo...

- La función con el nombre específico ACT_7 se elimina como candidato, porque el esquema "NERO" no está incluido en la vía de acceso de SQL.
- La función con el nombre específico ACT_3 se elimina como candidato, porque tiene el número incorrecto de parámetros. ACT_1 y ACT_6 se eliminan porque en ambos casos el primer argumento no se puede promocionar al tipo de datos del primer parámetro.
- Puesto que sigue habiendo más de un candidato, los argumentos se tienen en cuenta siguiendo un orden.
- Para el primer argumento, todas las funciones restantes — ACT_2, ACT_4 y ACT_5 coinciden exactamente con el tipo de argumento. No se puede pasar por alto ninguna de las funciones; así pues, examinemos el argumento siguiente.
- Para este segundo argumento, ACT_2 y ACT_5 coinciden exactamente, pero no sucede lo mismo con ACT_4, por lo cual queda descartado. Se examina el siguiente argumento para determinar alguna diferencia entre ACT_2 y ACT_5.
- En el tercer y último argumento, ni ACT_2 ni ACT_5 coinciden exactamente con el tipo de argumento, pero ambos son igualmente aceptables.

- Quedan dos funciones, ACT_2 y ACT_5, con firmas de parámetros idénticas. La criba final consiste en determinar qué esquema de la función aparece en primer lugar en la vía de acceso de SQL y, finalmente, se elige ACT_5 sobre esta base.

Invocación de función

Cuando ya se ha seleccionado la función, pueden darse todavía algunos motivos por los cuales no se pueda utilizar alguna función. Cada función se define para que devuelva un resultado con un tipo de datos específico. Si este tipo de datos resultante no es compatible con el contexto donde se invoca la función, se producirá un error. Por ejemplo, con las funciones denominadas STEP, en esta ocasión con tipos de datos diferentes como resultado:

```
STEP(SMALLINT) devuelve CHAR(5)
STEP(DOUBLE)   devuelve INTEGER
```

y la referencia de función siguiente (donde S es una columna SMALLINT):

```
SELECT ... 3 + STEP(S) ...
```

entonces, se elige el primer STEP, ya que los tipos de argumento coinciden plenamente. Se produce un error en la sentencia porque el tipo resultante es CHAR(5) en lugar de un tipo numérico, tal como necesita un argumento del operador de suma.

Otros ejemplos en los que puede ocurrir esto son los siguientes, ambos darán como resultado un error en la sentencia:

1. Se ha hecho referencia a la función en una cláusula FROM, pero la función seleccionada por el paso de resolución de función ha sido una función escalar ni de columna.
2. El caso contrario en el que el contexto llama a una función escalar o de columna y la resolución de función selecciona una función de tabla.

En los casos donde los argumentos de la invocación de función no coinciden exactamente con los tipos de datos de los argumentos de la función seleccionada, los argumentos se convierten al tipo de datos del parámetro durante la ejecución utilizando las mismas reglas que la asignación a las columnas (consulte el apartado "Asignaciones y comparaciones" en la página 105). También se incluye el caso en el que la precisión, escala o longitud difiere entre el argumento y el parámetro.

Métodos

Un método de base de datos de un tipo estructurado es una relación entre un conjunto de valores de datos de entrada y un conjunto de valores resultantes, donde el primer valor de entrada (o *argumento sujeto*) tiene el mismo tipo, o es un subtipo del tipo sujeto (también llamado *parámetro sujeto*) del método. Por ejemplo, un método llamado CITY, de tipo ADDRESS, puede recibir valores de datos de entrada de tipo VARCHAR y el resultado es un valor ADDRESS (o un subtipo de ADDRESS).

Los métodos se definen implícita o explícitamente, como parte de la definición de un tipo estructurado definido por el usuario.

Los métodos definidos implícitamente se crean para cada tipo estructurado. Se define un método observador para cada atributo del tipo estructurado. Los métodos observadores permiten que una aplicación obtenga el valor de un atributo para una instancia del tipo. También se definen métodos mutadores para cada atributo, que permiten que una aplicación cambie la instancia de tipo modificando el valor de un atributo de una instancia de tipo. El método CITY descrito anteriormente es un ejemplo de método mutador para el tipo ADDRESS.

Los métodos definidos explícitamente o *métodos definidos por el usuario* son métodos que se registran en el catálogo SYSCAT.FUNCTIONS de una base de datos, utilizando una combinación de las sentencias CREATE TYPE (o ALTER TYPE ADD METHOD) y CREATE METHOD. Todos los métodos definidos para un tipo estructurado se definen en el mismo esquema que el tipo.

Gracias a los métodos definidos por el usuario, DB2 permite a los usuarios y desarrolladores de aplicaciones ampliar la función del sistema de base de datos. Esto se consigue añadiendo definiciones de método, proporcionadas por usuarios o proveedores, que se aplican a instancias de tipo estructurado en el núcleo de la base de datos. Las definiciones de método añadidas proporcionan un mayor nivel de rendimiento, en comparación con la recuperación de filas de la base de datos y la aplicación de funciones sobre los datos recuperados. El definir métodos de base de datos también permite a la base de datos explotar los mismos métodos en el núcleo utilizado por una aplicación, lo que proporciona un mayor nivel de eficiencia en la interacción entre la aplicación y la base de datos. Esto da como resultado una mayor productividad para los desarrolladores de aplicaciones, pues es un enfoque más orientado a objetos.

Métodos definidos por el usuario: externos y SQL

Un método definido por el usuario puede ser externo o estar basado en una expresión SQL. Un método externo se define para la base de datos con una referencia a una biblioteca de códigos objeto y una función en dicha biblioteca que se ejecutará cuando se invoque el método. Un método basado en una

expresión SQL devuelve el resultado de la expresión SQL cuando se invoca el método. Tales métodos no necesitan ninguna biblioteca de códigos objeto, pues están escritos completamente en SQL.

Un método definido por el usuario puede devolver un solo valor cada vez que se invoca. Este valor puede ser un tipo estructurado. Un método se puede definir como *preservador del tipo* (utilizando SELF AS RESULT), para permitir que el tipo dinámico del argumento sujeto sea el tipo devuelto del método. Todos los métodos mutadores definidos implícitamente son preservadores del tipo.

Signaturas de método

Un método se identifica por su tipo sujeto, un nombre de método, el número de parámetros y los tipos de datos de sus parámetros. Esto es la signatura del método y debe ser exclusiva dentro de la base de datos.

Puede existir más de un método con el mismo nombre para un tipo estructurado, si se cumplen estas condiciones:

- el número de parámetros o los tipos de datos de los parámetros son diferentes
- no existe la misma signatura para un subtipo o supertipo del tipo sujeto del método
- no existe la misma signatura de función (utilizando el mismo tipo sujeto o cualquiera de sus subtipos o supertipos como primer parámetro).

Un nombre de método que tiene varias instancias de método se denomina *método sobrecargado*. Un nombre de método puede estar sobrecargado dentro de un tipo, lo que significa que existe más de un método de ese nombre para el tipo (todos los cuales tienen diferentes tipos de parámetros). Un nombre de método también puede estar sobrecargado en la jerarquía de tipos sujeto, en cuyo caso existe más de un método con ese nombre en la jerarquía de tipos, y estos métodos deben también tener tipos de parámetros diferentes.

Invocación de métodos

Un método se puede invocar haciendo referencia, en un contexto permitido, al nombre de método precedido por una referencia a una instancia de tipo estructurado (el argumento sujeto) y por el operador de doble punto (..). A continuación debe seguir la lista de argumentos entre paréntesis. El método que se invoca realmente se determina basándose en el tipo estático del tipo sujeto, utilizando la resolución de métodos descrita en la sección siguiente. Los métodos definidos con WITH FUNCTION ACCESS también se pueden invocar utilizando la invocación de funciones, en cuyo se aplican las reglas normales de la resolución de funciones.

Resolución de métodos

Al invocar un método, el gestor de bases de datos debe decidir cuál de los posibles métodos con el mismo nombre es el más apropiado. Las funciones (incorporadas o definidas por el usuario) no se tienen en cuenta durante la resolución del método.

Un argumento es un valor que se pasa a un método al invocarlo. Cuando un método se invoca en SQL, se le pasa el argumento sujeto (de algún tipo estructurado) y opcionalmente una lista de argumentos. Son argumentos posicionales en tanto que la semántica de dichos argumentos viene determinada por su posición en la lista de argumentos. El argumento sujeto se considera que es el primer argumento. Un parámetro es una definición formal de unos datos de entrada para un método.

Cuando se define un método para la base de datos, ya sea implícitamente (método generado por el sistema para un tipo) o un explícitamente (método definido por el usuario), se especifican sus parámetros (con el parámetro sujeto como primer parámetro) y el orden de sus definiciones determina sus posiciones y su semántica. Por tanto, cada parámetro es una entrada posicional determinada de un método. En la invocación, un argumento corresponde a un parámetro determinado en virtud a la posición que éste ocupe en la lista de argumentos.

El gestor de bases de datos utiliza el nombre de método proporcionado en la invocación, el número y los tipos de datos de los argumentos, todos los métodos que tienen el mismo nombre para el tipo estático del argumento sujeto y los tipos de datos de sus parámetros correspondientes como base para decidir si selecciona o no un método.

A continuación se muestran los resultados posibles del proceso de decisión:

1. Un método determinado se considera que es el más apropiado. Por ejemplo, para los métodos denominados RISK del tipo SITE con signaturas definidas como:

```
PROXIMITY(INTEGER) FOR SITE  
PROXIMITY(DOUBLE) FOR SITE
```

la siguiente invocación de método (donde ST es una columna SITE, DB es una columna DOUBLE):

```
SELECT ST..PROXIMITY(DB) ...
```

se elegiría el segundo PROXIMITY.

la siguiente invocación de método (donde SI es una columna SMALLINT):

```
SELECT ST..PROXIMITY(SI) ...
```


elegirá el primer PROXIMITY, pues SMALLINT se puede promover a INTEGER y es una coincidencia mejor que DOUBLE, que se encuentra más abajo en la lista de prioridad.

Cuando se tienen en cuenta argumentos que son tipos estructurados, la lista de prioridad incluye los supertipos del tipo estático del argumento. La función que mejor se ajusta es la definida con el parámetro de supertipo más cercano, en la jerarquía de tipos estructurados, al tipo estático del argumento de función.

2. Ningún método se considera una opción aceptable. Por ejemplo, para las dos funciones del caso anterior y la siguiente referencia a función (donde C es una columna CHAR(5)):

```
SELECT ST..PROXIMITY(C) ...
```

el argumento es incoherente con el parámetro de las dos funciones PROXIMITY.

3. Se selecciona un método determinado basándose en los métodos de la jerarquía de tipos y en el número y tipos de datos de los argumentos pasados en la invocación. Por ejemplo, para los métodos denominados RISK de los tipos SITE y DRILLSITE (un subtipo de SITE) con signaturas definidas como:

```
RISK(INTEGER) FOR DRILLSITE
RISK(DOUBLE) FOR SITE
```

la siguiente invocación de método (donde DRST es una columna DRILLSITE, DB es una columna DOUBLE):

```
SELECT DRST..RISK(DB) ...
```

se elegirá el segundo RISK, pues DRILLSITE se puede promocionar a SITE.

La siguiente referencia a método (donde SI es una columna SMALLINT):

```
SELECT DRST..RISK(SI) ...
```

elegirá el primer RISK, pues SMALLINT se puede promocionar a INTEGER, que está más cerca en la lista de prioridad que DOUBLE, y DRILLSITE es una opción mejor que SITE, que es un supertipo.

Los métodos con la misma jerarquía de tipos no pueden tener las mismas signaturas, considerando los parámetros que no sean el parámetro sujeto.

Método de elección de la mejor opción

La comparación de los tipos de datos de los argumentos con los tipos de datos definidos de los parámetros del método en cuestión es la base para

Métodos

decidir qué método de un grupo de métodos de igual nombre es el más apropiado. Observe que el tipo de datos del resultado del método en cuestión no interviene en esa decisión.

La resolución del método se realiza siguiendo los pasos siguientes.

1. En primer lugar, busque todos los métodos del catálogo (SYSCAT.FUNCTIONS) que cumplan las condiciones siguientes:
 - el nombre del método coincide con el nombre de invocación, y el parámetro sujeto es el mismo tipo o es un supertipo del tipo estático del argumento sujeto
 - el número de parámetros definidos coincide con la invocación
 - cada argumento de invocación coincide con el parámetro definido correspondiente del método en cuanto al tipo de datos o se puede "promocionar" a ese tipo (vea "Promoción de los tipos de datos" en la página 100).
2. A continuación, examine de izquierda a derecha cada argumento de la invocación del método. El argumento situado más a la izquierda (y por tanto el primer argumento) es el parámetro SELF implícito. Por ejemplo, un método definido para el tipo ADDRESS_T tiene un primer parámetro implícito de tipo ADDRESS_T.

Para cada argumento, elimine todas las funciones que no sean la mejor coincidencia para ese argumento. La mejor coincidencia para un argumento dado es el primer tipo de datos que aparece en la lista de prioridad, correspondiente al tipo de datos del argumento en la Tabla 5 en la página 100 para el cual existe una función con un parámetro de ese tipo de datos.

En esta comparación no se tienen en cuenta las longitudes, precisiones y escalas ni el atributo "FOR BIT DATA". Por ejemplo, un argumento DECIMAL(9,1) se considera una coincidencia exacta para un parámetro DECIMAL(6,5) mientras que un argumento VARCHAR(19) es una coincidencia exacta para un parámetro VARCHAR(6).

La mejor coincidencia para un argumento de tipo estructurado definido por el usuario es el propio argumento; la siguiente mejor coincidencia es el supertipo inmediato, y así sucesivamente para cada supertipo del argumento. Observe que sólo se tiene en cuenta el tipo estático (tipo declarado) del argumento de tipo estructurado, no el tipo dinámico (tipo más específico).

3. Como máximo, después del Paso 2 queda un método elegible. Este es el método que se elige.
4. Si después del Paso 2 no queda ningún método elegible, se produce un error (SQLSTATE 42884).

Ejemplo de resolución de método

A continuación se muestra un ejemplo de una resolución de método satisfactoria.

Existen siete métodos FOO para tres tipos estructurados definidos en una jerarquía de GOVERNOR como un subtipo de EMPEROR, como un subtipo de HEADOFSTATE, registrados con firmas:

```
CREATE METHOD FOO (CHAR(5), INT, DOUBLE) FOR HEADOFSTATE SPECIFIC FOO_1 ...
CREATE METHOD FOO (INT, INT, DOUBLE) FOR HEADOFSTATE SPECIFIC FOO_2 ...
CREATE METHOD FOO (INT, INT, DOUBLE, INT) FOR HEADOFSTATE SPECIFIC FOO_3 ...
CREATE METHOD FOO (INT, DOUBLE, DOUBLE) FOR EMPEROR SPECIFIC FOO_4 ...
CREATE METHOD FOO (INT, INT, DOUBLE) FOR EMPEROR SPECIFIC FOO_5 ...
CREATE METHOD FOO (SMALLINT, INT, DOUBLE) FOR EMPEROR SPECIFIC FOO_6 ...
CREATE METHOD FOO (INT, INT, DEC(7,2)) FOR GOVERNOR SPECIFIC FOO_7 ...
```

La referencia al método es la siguiente (donde I1 e I2 son columnas INTEGER, D es una columna DECIMAL y E es una columna EMPEROR):

```
SELECT E..FOO(I1, I2, D) ...
```

De acuerdo con el algoritmo...

FOO_7 se elimina como candidato, porque el tipo GOVERNOR es un subtipo de EMPEROR (no un supertipo).

FOO_3 se elimina como candidato, porque tiene un número erróneo de parámetros.

FOO_1 y FOO_6 se eliminan porque, en ambos casos, el primer argumento (no el argumento sujeto) no se puede promocionar al tipo de datos del primer parámetro. Puesto que sigue habiendo más de un candidato, los argumentos se tienen en cuenta siguiendo un orden.

Para el argumento sujeto, FOO_2 es un supertipo, mientras que FOO_4 y FOO_5 coinciden con el argumento sujeto.

Para el primer argumento, los métodos restantes, FOO_4 y FOO_5, coinciden exactamente con el tipo del argumento. No se puede descartar ningún método, por tanto se debe examinar el argumento siguiente.

Para este segundo argumento, FOO_5 es una coincidencia exacta, mientras que FOO_4 no lo es, por lo cual se descarta. Esto deja FOO_5 como método elegido.

Invocación de métodos

Una vez seleccionado el método, pueden todavía existir algunos motivos por los cuales no se pueda utilizar el método.

Cada método está definido para devolver un resultado con un tipo de datos específico. Si este tipo de datos resultante no es compatible con el contexto

Métodos

donde se invoca el método, se produce un error. Por ejemplo, suponga que se definen los siguientes métodos llamados STEP, cada uno con un tipo de datos diferentes como resultado:

```
STEP(SMALLINT) FOR TYPEA RETURNS CHAR(5)
STEP(DOUBLE) FOR TYPEA RETURNS INTEGER
```

y la siguiente referencia a método (donde S es una columna SMALLINT y TA es una columna de tipo TYPEA):

```
SELECT 3 + TA..STEP(S) ...
```

en este caso se elige el primer STEP, pues hay una coincidencia exacta del tipo del argumento. Se produce un error en la sentencia, porque el tipo resultante es CHAR(5) en lugar de un tipo numérico, tal como necesita un argumento del operador de suma.

Observe que cuando el método seleccionado es un método conservador del tipo:

- el tipo estático resultante tras la resolución de la función es el mismo que el tipo estático del argumento sujeto de la invocación del método
- el tipo dinámico resultante cuando se invoca el método es el mismo que el tipo dinámico del argumento sujeto de la invocación del método.

Esto puede ser un subtipo del tipo resultante especificado en la definición del método conservador del tipo, que a su vez puede ser un supertipo del tipo dinámico devuelto realmente cuando se procesa el método.

En los casos donde los argumentos de la invocación del método no coinciden exactamente con los tipos de datos de los parámetros del método seleccionado, los argumentos se convierten al tipo de datos del parámetro durante la ejecución, utilizando las mismas reglas que para la asignación a columnas (vea “Asignaciones y comparaciones” en la página 105). Esto incluye el caso en el que la precisión, escala o longitud difiere entre el argumento y el parámetro, pero excluye el caso en el que el tipo dinámico del argumento es un subtipo del tipo estático del parámetro.

Semántica de enlace conservador

En una base de datos se producen situaciones en las que las funciones, los métodos y los tipos de datos se resuelven cuando se procesa la sentencia y el gestor de bases de datos debe poder repetir esta resolución. Esto es aplicable a:

- sentencias DML estáticas de paquetes,
- vistas,
- desencadenantes,
- restricciones de comprobación y

- rutinas SQL.

En el caso de las sentencias DML estáticas de paquetes, las referencias a funciones, métodos y tipos de datos se resuelven durante la operación de enlace. Las referencias a funciones, métodos y tipos de datos en las vistas, desencadenantes y restricciones de comprobación se resuelven cuando se crea el objeto de base de datos.

Si la resolución de la función o método se ejecuta de nuevo para cualquier referencia a función o método de estos objetos, el comportamiento podría cambiar si se ha añadido una nueva función o método con una signatura que coincide mejor, pero el ejecutable real realiza operaciones diferentes. Similarmente, si la resolución se ejecuta de nuevo para cualquier tipo de datos de estos objetos, podría cambiar el comportamiento si se ha añadido un nuevo tipo de datos con el mismo nombre en un esquema diferente que también está en la vía de acceso de SQL. Para evitar esto, donde sea necesario, el gestor de bases de datos aplica el concepto de *semántica de enlace conservador*. Este concepto asegura que las referencias a función y al tipo de datos se resuelvan utilizando la misma vía de acceso de SQL que cuando se enlazó. Además, la indicación horaria de la creación de funciones ²⁶, métodos y tipos de datos considerados durante la resolución no es posterior a la hora en la que se enlazó la sentencia ²⁷. De esta forma, sólo se considerarán las funciones y tipos de datos que se tuvieron en cuenta durante la resolución de funciones, métodos y tipos de datos cuando se procesó originalmente la sentencia. Por tanto, las funciones, métodos y tipos de datos recién creados no se tienen en cuenta cuando se aplica la semántica de enlace conservador.

En el caso de los paquetes DML estáticos, los paquetes se pueden volver a enlazar implícitamente o explícitamente emitiendo el mandato REBIND (o bien la API correspondiente) o bien el mandato BIND (o la API correspondiente). El reenlace implícito se ejecuta siempre para resolver funciones, métodos y tipos de datos con la semántica de enlace conservador. El mandato REBIND proporciona la posibilidad de resolver con la semántica de enlace conservador (opción RESOLVE CONSERVATIVE) o resolver teniendo en cuenta las funciones, métodos y tipos de datos recién creados (resolución por omisión o mediante la opción RESOLVE ANY).

26. A partir de la Versión 6.1, las funciones incorporadas añadidas tienen una indicación horaria basada en la hora en la que se creó o migró la base de datos.

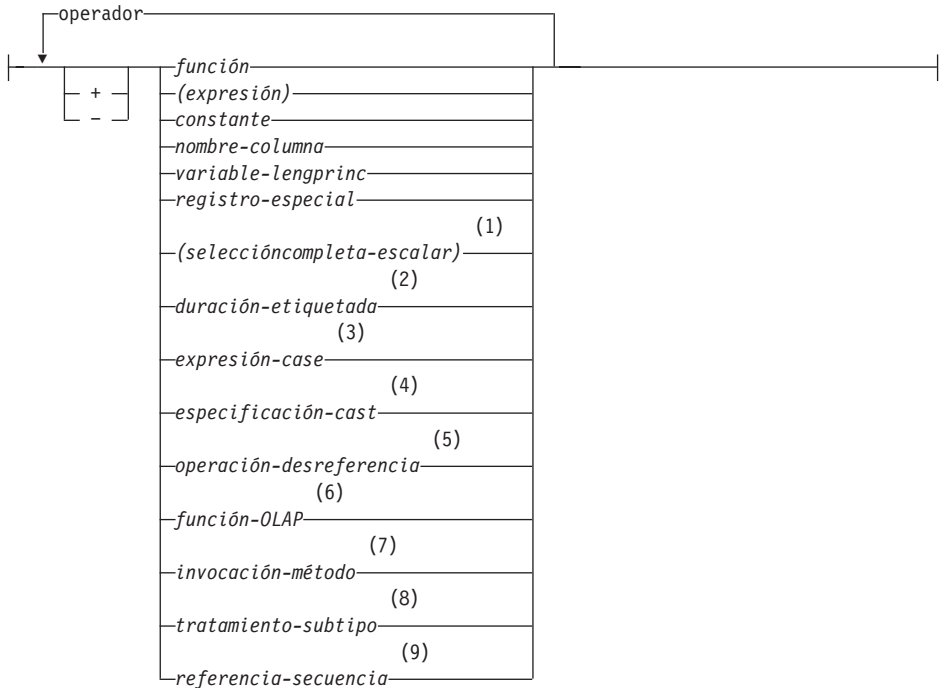
27. Para las vistas, el enlace conservador también asegura que las columnas de la vista sean las mismas que las existentes cuando se creó la vista. Por ejemplo, una vista definida mediante el asterisco en la lista de selección no tendrá en cuenta ninguna columna añadida a las tablas subyacentes una vez se ha creado la vista.

Expresiones

Expresiones

Una expresión especifica un valor. Puede ser un valor simple, formado sólo por una constante o un nombre de columna, o puede ser más complejo. Si se utilizan repetidamente expresiones similares complejas, debe considerarse la posibilidad de utilizar una función SQL para encapsular una expresión común. Vea “CREATE FUNCTION (SQL, escalar, de tabla o de fila)” en la página 761 para obtener más información.

expresión:



operador:



Notas:

- 1 Vea “Selección completa escalar” en la página 184 para obtener más información.
- 2 Vea “Duraciones etiquetadas” en la página 184 para obtener más información.
- 3 Vea “Expresiones CASE” en la página 190 para obtener más información.
- 4 Vea “Especificaciones CAST” en la página 193 para obtener más información.
- 5 Vea “Operaciones de desreferencia” en la página 196 para obtener más información.
- 6 Vea “Funciones OLAP” en la página 197 para obtener más información.
- 7 Vea “Invocación de métodos” en la página 204 para obtener más información.
- 8 Vea “Tratamiento de los subtipos” en la página 206 para obtener más información.
- 9 Vea “Referencia de secuencia” en la página 207 para obtener más información.
- 10 || se utiliza como sinónimo de CONCAT.

Sin operadores

Si no se utilizan operadores, el resultado de la expresión es el valor especificado.

Ejemplos: `SALARY` `:SALARY` `'SALARY'` `MAX(SALARY)`

Con el operador de concatenación

El operador de concatenación (CONCAT) enlaza dos operandos de serie para formar una *expresión de serie*.

Los operandos de la concatenación deben ser series compatibles. Es preciso tener en cuenta que una serie binaria no se puede concatenar con una serie de caracteres, incluso las que se definen como FOR BIT DATA (SQLSTATE 42884). Para obtener más información sobre la compatibilidad, consulte la matriz de compatibilidad de la Tabla 7 en la página 105.

Si ambos operandos pueden ser nulos, el resultado también podrá ser nulo; si alguno de ellos es nulo, el resultado es el valor nulo. De lo contrario, el resultado constará de la serie del primer operando seguida por la del segundo. La comprobación se efectúa para detectar los datos mixtos formados defectuosamente al realizar la concatenación.

Expresiones

La longitud del resultado es la suma de las longitudes de los operandos.

El tipo de datos y el atributo de longitud del resultado vienen determinados por los de los operandos, tal como se muestra en la tabla siguiente:

Tabla 10. Tipo de datos y longitud de los operandos concatenados

Operandos	Atributos de longitud combinados	Resultado
CHAR(A) CHAR(B)	<255	CHAR(A+B)
CHAR(A) CHAR(B)	>254	VARCHAR(A+B)
CHAR(A) VARCHAR(B)	<4001	VARCHAR(A+B)
CHAR(A) VARCHAR(B)	>4000	LONG VARCHAR
CHAR(A) LONG VARCHAR	-	LONG VARCHAR
VARCHAR(A) VARCHAR(B)	<4001	VARCHAR(A+B)
VARCHAR(A) VARCHAR(B)	>4000	LONG VARCHAR
VARCHAR(A) LONG VARCHAR	-	LONG VARCHAR
LONG VARCHAR LONG VARCHAR	-	LONG VARCHAR
CLOB(A) CHAR(B)	-	CLOB(MIN(A+B, 2G))
CLOB(A) VARCHAR(B)	-	CLOB(MIN(A+B, 2G))
CLOB(A) LONG VARCHAR	-	CLOB(MIN(A+32K, 2G))
CLOB(A) CLOB(B)	-	CLOB(MIN(A+B, 2G))
GRAPHIC(A) GRAPHIC(B)	<128	GRAPHIC(A+B)
GRAPHIC(A) GRAPHIC(B)	>127	VARGRAPHIC(A+B)
GRAPHIC(A) VARGRAPHIC(B)	<2001	VARGRAPHIC(A+B)
GRAPHIC(A) VARGRAPHIC(B)	>2000	LONG VARGRAPHIC
GRAPHIC(A) LONG VARGRAPHIC	-	LONG VARGRAPHIC
VARGRAPHIC(A) VARGRAPHIC(B)	<2001	VARGRAPHIC(A+B)
VARGRAPHIC(A) VARGRAPHIC(B)	>2000	LONG VARGRAPHIC
VARGRAPHIC(A) LONG VARGRAPHIC	-	LONG VARGRAPHIC

Tabla 10. Tipo de datos y longitud de los operandos concatenados (continuación)

Operandos	Atributos de longitud combinados	Resultado
LONG VARGRAPHIC LONG VARGRAPHIC	-	LONG VARGRAPHIC
DBCLOB(A) GRAPHIC(B)	-	DBCLOB(MIN(A+B, 1G))
DBCLOB(A) VARGRAPHIC(B)	-	DBCLOB(MIN(A+B, 1G))
DBCLOB(A) LONG VARGRAPHIC	-	DBCLOB(MIN(A+16K, 1G))
DBCLOB(A) DBCLOB(B)	-	DBCLOB(MIN(A+B, 1G))
BLOB(A) BLOB(B)	-	BLOB(MIN(A+B, 2G))

Observe que, para que haya compatibilidad con las versiones anteriores, no hay escalada automática de los resultados que implica tipos de datos LONG a los tipos de datos LOB. Por ejemplo, la concatenación de un valor CHAR(200) y un valor LONG VARCHAR totalmente completo da como resultado un error en lugar de una promoción de un tipo de datos CLOB.

La página de códigos del resultado se considera una página de códigos derivada que viene determinada por la página de códigos de sus operandos tal como se describe en el apartado “Reglas para las conversiones de series” en la página 124.

Un operando puede ser un marcador de parámetros. Si se utiliza un marcador de parámetros, el tipo de datos y los atributos de longitud del operando se consideran los mismos que los del operando que no es el marcador de parámetros. El orden de las operaciones tiene su importancia, puesto que determina estos atributos en casos en los que se produce una concatenación anidada.

Ejemplo 1: Si FIRSTNAME es Pierre y LASTNAME es Fermat, entonces lo siguiente:

```
FIRSTNAME CONCAT ' ' CONCAT
LASTNAME
```

devuelve el valor Pierre Fermat

Ejemplo 2: Dado:

- COLA definido como VARCHAR(5) con valor 'AA'
- :host_var definida como una variable del lenguaje principal con una longitud 5 y el valor 'BB '

Expresiones

- COLC definido como CHAR(5) con valor 'CC'
- COLD definido como CHAR(5) con valor 'DDDDD'

El valor de: COLA **CONCAT** :host_var **CONCAT** COLC **CONCAT** COLD es:

```
'AABB  CC  DDDDD'
```

El tipo de datos es VARCHAR, el atributo de longitud es 17 y la página de códigos resultante es la página de códigos de la base de datos.

Ejemplo 3: Dado:

COLA se define como CHAR(10)

COLB se define como VARCHAR(5)

El marcador de parámetros de la expresión:

```
COLA CONCAT COLB CONCAT ?
```

se considera VARCHAR(15), ya que COLA **CONCAT** COLB se evalúa primero dando como resultado el primer operando de la segunda operación **CONCAT**.

Tipos definidos por el usuario

No se puede utilizar un tipo definido por el usuario con el operador de concatenación, aunque sea un tipo diferenciado con un tipo de datos fuente de tipo serie. Para poder concatenar, es preciso crear una función con el operador **CONCAT** como fuente. Por ejemplo, si existieran los tipos diferenciados **TITLE** y **TITLE_DESCRIPTION**, ambos con tipos de datos **VARCHAR(25)**, la siguiente función definida por el usuario, **ATTACH**, se podría utilizar para concatenarlos.

```
CREATE FUNCTION ATTACH (TITLE, TITLE_DESCRIPTION)  
RETURNS VARCHAR(50) SOURCE CONCAT (VARCHAR(), VARCHAR())
```

También existe la posibilidad de sobrecargar el operador de concatenación empleando una función definida por el usuario para añadir los tipos de datos nuevos.

```
CREATE FUNCTION CONCAT (TITLE, TITLE_DESCRIPTION)  
RETURNS VARCHAR(50) SOURCE CONCAT (VARCHAR(), VARCHAR())
```

Con operadores aritméticos

Si se utilizan operadores aritméticos, el resultado de la expresión es un valor derivado de la aplicación de los operadores a los valores de los operandos.

Si cualquier operando puede ser nulo o la base de datos está configurada con **DFT_SQLMATHWARN** establecido en sí, el resultado puede ser nulo.

Si algún operando tiene el valor nulo, el resultado de la expresión es el valor nulo.

Los operadores numéricos se pueden aplicar a tipos numéricos con signo y a tipos de fecha y hora (vea “Aritmética de fecha y hora en SQL” en la página 185). Por ejemplo, `USER+2` no es válido. Las funciones derivadas se pueden definir para operaciones aritméticas sobre tipos diferenciados con un tipo fuente que sea un tipo numérico con signo.

El operador de prefijo `+` (más unitario) no modifica su operando. El operador de prefijo `-` (menos unitario) invierte el signo de un operando distinto de cero y, si el tipo de datos de `A` es un entero pequeño, el tipo de datos de `-A` es un entero grande. El primer carácter del símbolo que sigue a un operador de prefijo no debe ser un signo más ni un signo menos.

Los *operadores infijos* `+`, `-`, `*` y `/` especifican suma, resta, multiplicación y división respectivamente. El valor del segundo operando de una división no debe ser cero. Estos operadores también se pueden tratar como funciones. Por consiguiente, la expresión `+(a,b)` es equivalente a la función de “operador” cuya expresión es `a+b`.

Errores aritméticos

Si se produce un error aritmético como, por ejemplo, una división por cero o se produce un desbordamiento numérico durante el proceso de una expresión, se devuelve un error y la sentencia de SQL que procesa la expresión falla con un error (SQLSTATE 22003 ó 22012).

Una base de datos puede configurarse (utilizando `DFT_SQLMATHWARN` establecido en sí) para que los errores aritméticos devuelvan un valor nulo para la expresión, emitan un aviso (SQLSTATE 01519 ó 01564) y prosigan con el proceso de la sentencia de SQL. Cuando los errores aritméticos se tratan como nulos, hay implicaciones en los resultados de las sentencias de SQL. A continuación encontrará algunos ejemplos de dichas implicaciones.

- Un error aritmético que se produce en la expresión que es el argumento de una función de columna provoca que se ignore la fila en la determinación del resultado de la función de columna. Si el error aritmético ha sido un desbordamiento, puede afectar de manera significativa a los valores del resultado.
- Un error aritmético que se produce en la expresión de un predicado en una cláusula `WHERE` puede hacer que no se incluyan filas en el resultado.
- Un error aritmético que se produce en la expresión de un predicado en una restricción de comprobación da como resultado el proceso de actualización o inserción ya que la restricción no es falsa.

Si estos tipos de efectos no son aceptables, deben seguirse pasos adicionales para manejar el error aritmético y producir resultados aceptables. Algunos ejemplos son:

Expresiones

- añadir una expresión case para comprobar la división por cero y establecer el valor deseado para dicha situación
- añadir predicados adicionales para manejar los nulos (por ejemplo, una restricción de comprobación en columnas sin posibilidad de nulos daría:
`check (c1*c2 is not null and c1*c2>5000)`

para hacer que la restricción se violase en un desbordamiento).

Dos operandos enteros

Si ambos operandos de un operador aritmético son enteros, la operación se realiza en binario y el resultado es un *entero grande* a no ser que uno de los operandos (o ambos) sea un entero grande, en cuyo caso el resultado es un entero grande. Se pierde cualquier resto de una división. El resultado de una operación aritmética de enteros (incluyendo el menos unitario) debe estar dentro del rango del tipo del resultado.

Operandos enteros y decimales

Si un operando es un entero y el otro es un decimal, la operación se realiza en decimal utilizando una copia temporal del entero que se habrá convertido a número decimal con la precisión p y la escala 0. p es 19 para un entero superior, 11 para un entero grande y 5 para un entero pequeño.

Dos operandos decimales

Si los dos operandos son decimales, la operación se efectúa en decimal. El resultado de cualquier operación aritmética decimal es un número decimal con una precisión y una escala que dependen de la operación y de la precisión y la escala de los operandos. Si la operación es una suma o una resta y los operandos no tienen la misma escala, la operación se efectúa con una copia temporal de uno de los operandos. La copia del operando más corto se extiende con ceros de cola de manera que la parte de la fracción tenga el mismo número de dígitos que el otro operando.

El resultado de una operación decimal no debe tener una precisión mayor que 31. El resultado de una suma, resta y multiplicación decimal se obtiene de un resultado temporal que puede tener una precisión mayor que 31. Si la precisión del resultado temporal no es mayor que 31, el resultado final es el mismo que el resultado temporal.

Aritmética decimal en SQL

Las fórmulas siguientes definen la precisión y escala del resultado de operaciones decimales en SQL. Los símbolos p y s indican la precisión y la escala del primer operando y los símbolos p' y s' indican y la precisión y la escala del segundo operando.

Suma y resta

La precisión es $\min(31, \max(p-s, p'-s') + \max(s, s') + 1)$. La escala del resultado de una suma o una resta es $\max(s, s')$.

Multiplicación

La precisión del resultado de una multiplicación es $\min(31, p+p')$ y la escala es $\min(31, s+s')$.

División

La precisión del resultado de la división es 31. La escala es $31-p+s'$. La escala no debe ser negativa.

Nota: El parámetro de configuración de base de datos `MIN_DEC_DIV_3` modifica la escala para las operaciones aritméticas decimales que incluyen la división. Si el valor del parámetro se establece en `NO`, la escala se calcula como $31-p+s'$. Si el parámetro se establece en `YES`, la escala se calcula como $\text{MAX}(3, 31-p+s')$. Esto asegura que el resultado de una división decimal tenga siempre una escala de 3 como mínimo (la precisión es siempre 31). Consulte la publicación *Administration Guide* para obtener información adicional relacionada con el parámetro de configuración de base de datos `MIN_DEC_DIV_3`.

Operandos de coma flotante

Si cualquiera de los dos operandos de un operador aritmético es de coma flotante, la operación se realiza en coma flotante, convirtiendo primero los operandos a números de coma flotante de doble precisión, si es necesario. Por lo tanto, si cualquier elemento de una expresión es un número de coma flotante, el resultado de la expresión es un número de coma flotante de precisión doble.

Una operación en la que intervenga un número de coma flotante y un entero se realiza con una copia temporal del entero que se ha convertido a coma flotante de precisión doble. Una operación en la que intervenga un número de coma flotante y un número decimal se efectúa con una copia temporal del número decimal que se ha convertido a coma flotante de precisión doble. El resultado de una operación de coma flotante debe estar dentro del rango de los números de coma flotante.

Tipos definidos por el usuario como operandos

Un tipo definido por el usuario no puede utilizarse con operadores aritméticos incluso si su tipo de datos fuente es numérico. Para llevar a cabo una operación aritmética, cree una función con el operador aritmético como fuente. Por ejemplo, si existen los tipos diferenciados `INCOME` y `EXPENSES`, y ambos tienen tipos de datos `DECIMAL(8,2)`, se podría utilizar la función `REVENUE` definida por el usuario para restar uno de otro, de la forma siguiente:

```
CREATE FUNCTION REVENUE (INCOME, EXPENSES)
RETURNS DECIMAL(8,2) SOURCE "-" (DECIMAL, DECIMAL)
```

El operador - (menos) se puede sobrecargar de forma alternativa utilizando la función definida por el usuario para restar los tipos de datos nuevos.

Expresiones

```
CREATE FUNCTION "-" (INCOME, EXPENSES)
  RETURNS DECIMAL(8,2) SOURCE "-" (DECIMAL, DECIMAL)
```

Selección completa escalar

La *selección completa escalar*, tal como se utiliza en una expresión, es una selección completa, entre paréntesis, que devuelve una fila individual formada por un solo valor de columna. Si la selección completa no devuelve una fila, el resultado de la expresión es el valor nulo. Si el elemento de la lista de selección es una expresión que simplemente es un nombre de columna o una operación de desreferencia, el nombre de columna del resultado está basado en el nombre de la columna. Vea “selección completa” en la página 512 para obtener más información.

Operaciones de fecha y hora y duraciones

Los valores de indicación de fecha y hora se pueden aumentar, disminuir y restar. Estas operaciones pueden incluir números decimales llamados *duraciones*. A continuación se muestra una definición de las duraciones y una especificación de las reglas para la aritmética de la fecha y hora.

Una duración es un número que representa un intervalo de tiempo. Hay cuatro tipos de duraciones:

Duraciones etiquetadas

duración-etiquetada:

función	YEAR
(expresión)	YEARS
constante	MONTH
nombre-columna	MONTHS
variable-lengprinc	DAY
	DAYS
	HOUR
	HOURS
	MINUTE
	MINUTES
	SECOND
	SECONDS
	MICROSECOND
	MICROSECONDS

Una *duración etiquetada* representa una unidad de tiempo específica expresada por un número (que puede ser el resultado de una expresión) seguido de una de las siete palabras clave de duración: YEARS, MONTHS, DAYS, HOURS, MINUTES, SECONDS o MICROSECONDS.²⁸ El número especificado se

28. Tenga en cuenta que también se acepta la forma singular de estas palabras clave: YEAR, MONTH, DAY, HOUR, MINUTE, SECOND y MICROSECOND.

convierte como si se asignara a un número DECIMAL(15,0). Sólo puede utilizarse una duración etiquetada como operando de un operador aritmético en el que el otro operando sea un valor de tipo de datos DATE, TIME o TIMESTAMP. Así pues, la expresión HIREDATE + 2 MONTHS + 14 DAYS es válida, mientras que la expresión HIREDATE + (2 MONTHS + 14 DAYS) no lo es. En ambas expresiones, las duraciones etiquetadas son 2 MONTHS (meses) y 14 DAYS (días).

Duración de fecha

Una *duración de fecha* representa un número de años, meses y días, expresados como un número DECIMAL(8,0). Para interpretarse correctamente, el número debe tener el formato *aaaammdd.*, donde *aaaa* representa el número de años, *mm* el número de meses y *dd* el número de días.²⁹ El resultado de restar un valor de fecha de otro, como sucede en la expresión HIREDATE – BRTHDATE, es una duración expresada en fechas.

Duración de hora

Una *duración de hora* representa un número de horas, minutos y segundos, expresado como un número DECIMAL(6,0). Para interpretarse correctamente, el número debe tener el formato *hhmmss.*, donde *hh* representa el número de horas, *mm* el número de minutos y *ss* el número de segundos.²⁹ El resultado de restar un valor de hora de otro es una duración expresada en horas.

Duración de indicación de fecha y hora

Una *duración de fechas y hora* representa un número de años, meses, días, horas, minutos, segundos y microsegundos expresado como un número DECIMAL(20,6). Para interpretarse correctamente, el número debe tener el formato *aaaammddhhmmss.zzzzzz*, donde *aaaa*, *mm*, *dd*, *hh*, *mm*, *ss* y *zzzzzz* representan el número de años, meses, días, horas, minutos, segundos y microsegundos respectivamente. El resultado de restar un valor de indicación de fecha y hora de otro es una duración de indicación de fecha y hora.

Aritmética de fecha y hora en SQL

Las únicas operaciones aritméticas que pueden efectuarse con valores de indicación de fecha y hora son la suma y la resta. Si un valor de indicación de fecha y hora es un operando de suma, el otro operando debe ser una duración. A continuación, encontrará las reglas específicas que rigen la utilización del operador de suma con valores de indicación de fecha y hora.

- Si un operando es una fecha, el otro operando debe ser una duración de fecha o una duración etiquetada de YEARS, MONTHS o DAYS.
- Si un operando es una hora, el otro operando debe ser una duración de hora o una duración etiquetada de HOURS, MINUTES o SECONDS.

29. El punto en el formato indica un tipo de datos DECIMAL.

Expresiones

- Si un operando es una fecha y hora, el otro operando debe ser una duración. Cualquier tipo de duración es válido.
- Ningún operando del operador de suma puede ser un marcador de parámetros.

Las normas para la utilización del operador de resta con valores de indicación de fecha y hora no son las mismas que para la suma, porque un valor de indicación de fecha y hora no puede restarse de una duración, y porque la operación de restar dos valores de indicación de fecha y hora no es la misma que la operación de restar una duración de un valor de indicación de fecha y hora. A continuación se muestran las normas específicas que rigen la utilización del operador de resta con valores de indicación de fecha y hora.

- El primer operando es una fecha, el segundo operando debe ser una fecha, una duración de fecha, una representación de una fecha en forma de serie o una duración etiquetada de YEARS, MONTHS o DAYS.
- Si el segundo operando es una fecha, el primer operando debe ser una fecha o una representación de una fecha en forma de serie.
- Si el primer operando es una hora, el segundo operando debe ser una hora, una duración de hora, una representación de una hora en forma de serie o una duración etiquetada de HOURS, MINUTES o SECONDS.
- Si el segundo operando es una hora, el primer operando debe ser una hora o una representación de una hora en forma de serie.
- Si el primer operando es una fecha y hora, el segundo operando debe ser una fecha y hora, una representación de una fecha y hora en forma de serie o una duración.
- Si el segundo operando es una fecha y hora, el primer operando debe ser una fecha y hora o una representación de una fecha y hora en forma de serie.
- Ningún operando del operador de resta puede ser un marcador de parámetros.

Aritmética de fecha

Las fechas se pueden restar, aumentar o disminuir.

Resta de fechas: Al restar un fecha (DATE2) de otra (DATE1) se obtiene como resultado una duración de fecha que especifica el número de años, meses y días entre las dos fechas. El tipo de datos del resultado es DECIMAL(8,0). Si DATE1 es mayor o igual que DATE2, DATE2 se resta de DATE1. Si DATE1 es menor que DATE2, DATE1 se resta de DATE2 y el signo del resultado se convierte en negativo. La descripción siguiente clarifica los pasos que intervienen en el resultado de la operación = DATE1 - DATE2.

Si $\text{DAY}(\text{DATE2}) \leq \text{DAY}(\text{DATE1})$
entonces $\text{DAY}(\text{RESULT}) = \text{DAY}(\text{DATE1}) - \text{DAY}(\text{DATE2})$.

Si $\text{DAY}(\text{DATE2}) > \text{DAY}(\text{DATE1})$
entonces $\text{DAY}(\text{RESULT}) = N + \text{DAY}(\text{DATE1}) - \text{DAY}(\text{DATE2})$
donde $N = \text{el último día de MONTH}(\text{DATE2})$.
 $\text{MONTH}(\text{DATE2})$ se aumenta en 1.

Si $\text{MONTH}(\text{DATE2}) \leq \text{MONTH}(\text{DATE1})$
entonces $\text{MONTH}(\text{RESULT}) = \text{MONTH}(\text{DATE1}) - \text{MONTH}(\text{DATE2})$.

Si $\text{MONTH}(\text{DATE2}) > \text{MONTH}(\text{DATE1})$
entonces $\text{MONTH}(\text{RESULT}) = 12 + \text{MONTH}(\text{DATE1}) - \text{MONTH}(\text{DATE2})$.
 $\text{YEAR}(\text{DATE2})$ se aumenta en 1.

$\text{YEAR}(\text{RESULT}) = \text{YEAR}(\text{DATE1}) - \text{YEAR}(\text{DATE2})$.

Por ejemplo, el resultado de $\text{DATE}('15/3/2000') - '31/12/1999'$ es 00000215. (o una duración de 0 años, 2 meses y 15 días).

Incremento y disminución de fechas: Al añadir o restar una duración a una fecha se obtiene como resultado también una fecha. (En esta operación, un mes equivale a una página de un calendario. La adición de meses a una fecha es como ir pasando páginas a un calendario, empezando por la página en la que aparece la fecha.) El resultado debe estar comprendido entre las fechas 1 de enero de 0001 y 31 de diciembre de 9999, ambos inclusive.

Si se suma o resta una duración de años, solamente la parte de la fecha correspondiente a los años se verá afectada. Tanto el mes como el día permanecen inalterados, a no ser que el resultado fuera el 29 de febrero en un año no bisiesto. En este caso, el día se cambia a 28 y se define un indicador de aviso en la SQLCA para indicar el ajuste.

Del mismo modo, si se suma o resta una duración de meses, solamente los meses, y los años si fuera necesario, se verán afectados. La parte de una fecha correspondiente a los años no se cambia a no ser que el resultado no fuera válido (31 de setiembre, por ejemplo). En este caso, el día se establece en el último día del mes y se define un indicador de aviso en la SQLCA para indicar el ajuste.

Al añadir o restar una duración de días afectará, obviamente, a la parte de la fecha correspondiente a los días y potencialmente al mes y al año.

Las duraciones de fecha, ya sean positivas o negativas, también se pueden añadir y restar a las fechas. Tal como ocurre con las duraciones etiquetadas, se obtiene como resultado una fecha válida y se define un indicador de aviso en la SQLCA siempre que se deba efectuar un ajuste de fin de mes.

Cuando se suma una duración de fecha positiva a una fecha, o una duración de fecha negativa se resta de una fecha, la fecha aumenta el número especificado de años, meses y días, en ese orden. Así pues, $\text{DATE1} + X$, donde X es un número $\text{DECIMAL}(8,0)$ positivo, equivale a la expresión:

Expresiones

$DATE1 + YEAR(X) YEARS + MONTH(X) MONTHS + DAY(X) DAYS.$

Cuando una duración de fecha positiva se resta de una fecha, o bien se añade una duración de fecha negativa a una fecha, la fecha disminuye en el número días, meses y años especificados, en este orden. Así pues, $DATE1 - X$, donde X es un número DECIMAL(8,0) positivo, equivale a la expresión:

$DATE1 - DAY(X) DAYS - MONTH(X) MONTHS - YEAR(X) YEARS.$

Al añadir duraciones a fechas, la adición de un mes a una fecha determinada da la misma fecha un mes posterior a menos que la fecha no exista en el siguiente mes. En este caso, se establece la fecha correspondiente al último día del siguiente mes. Por ejemplo, 28 de enero más un mes da como resultado 28 de febrero y si se añade un mes al 29, 30 ó 31 de enero también se obtendrá como resultado el 28 de febrero o bien 29 de febrero si se trata de un año bisiesto.

Nota: Si se añade uno o más meses a una fecha determinada y del resultado se resta la misma cantidad de meses, la fecha final no tiene por qué ser necesariamente la misma que la original.

Aritmética de la hora

Las horas se pueden restar, aumentar o disminuir.

Resta de horas: El resultado de restar una hora (HOUR2) de otra (HOUR1) es una duración que especifica el número de horas, minutos y segundos entre las dos horas. El tipo de datos del resultado es DECIMAL(6,0).

Si HOUR1 es mayor o igual que HOUR2, HOUR2 se resta de HOUR1.

Si HOUR1 es menor que HOUR2, HOUR1 se resta de HOUR2 y el signo del resultado se convierte en negativo. La descripción siguiente clarifica los pasos que intervienen en el resultado de la operación = HOUR1 - HOUR2.

Si $SECOND(TIME2) \leq SECOND(TIME1)$
entonces $SECOND(RESULT) = SECOND(HOUR1) - SECOND(HOUR2).$

Si $SECOND(TIME2) > SECOND(TIME1)$
entonces $SECOND(RESULT) = 60 + SECOND(HOUR1) - SECOND(HOUR2).$
MINUTE(HOUR2) se aumenta entonces en 1.

Si $MINUTE(TIME2) \leq MINUTE(TIME1)$
entonces $MINUTE(RESULT) = MINUTE(HOUR1) - MINUTE(HOUR2).$

Si $MINUTE(TIME1) > MINUTE(TIME1)$
entonces $MINUTE(RESULT) = 60 + MINUTE(HOUR1) - MINUTE(HOUR2).$
HOUR(HOUR2) se aumenta entonces en 1.

$HOUR(RESULT) = HOUR(HOUR1) - HOUR(HOUR2).$

Por ejemplo, el resultado de $TIME('11:02:26') - '00:32:56'$ es 102930. (una duración de 10 horas, 29 minutos y 30 segundos).

Incremento y disminución de horas: El resultado de sumar una duración a una hora, o de restar una duración de una hora, es una hora. Se rechaza cualquier desbordamiento o subdesbordamiento de horas, garantizando de este modo que el resultado sea siempre una hora. Si se suma o resta una duración de horas, sólo se ve afectada la parte correspondiente a las horas. Los minutos y los segundos no cambian.

De manera parecida, si se suma o resta una duración de minutos, sólo se afecta a los minutos y, si fuera necesario, a las horas. La parte correspondiente a los segundos no cambia.

Al añadir o restar una duración de segundos afectará, obviamente, a la parte de la fecha correspondiente a los segundos y potencialmente a los minutos y a las horas.

Las duraciones de hora, tanto positivas como negativas, pueden también sumarse y restarse a las horas. El resultado es una hora que se ha incrementado o disminuido en el número especificado de horas, minutos y segundos, por ese orden. $\text{TIME1} + X$, donde "X" es un número DECIMAL(6,0), es equivalente a la expresión:

$$\text{TIME1} + \text{HOUR}(X) \text{ HOURS} + \text{MINUTE}(X) \text{ MINUTES} + \text{SECOND}(X) \text{ SECONDS}$$

Nota: Aunque la hora '24:00:00' se acepta como una hora válida, no se devuelve nunca como resultado de una suma o resta de horas, incluso si el operando de duración es cero (por ejemplo, hora('24:00:00')±0 segundos = '00:00:00').

Aritmética de la indicación de fecha y hora

Las indicaciones de fecha y hora se pueden restar, incrementar o disminuir.

Resta de indicaciones de fecha y hora: El resultado de restar una indicación de fecha y hora (TS2) de otra (TS1) es una duración de indicación de fecha y hora que especifica el número de años, meses, días, horas, minutos, segundos y microsegundos entre las dos indicaciones de fecha y hora. El tipo de datos del resultado es DECIMAL(20,6).

Si TS1 es mayor o igual que TS2, TS2 se resta de TS1. Si TS1 es menor que TS2, TS1 se resta de TS2 y el signo del resultado se convierte en negativo. La descripción siguiente clarifica los pasos que intervienen en el resultado de la operación = TS1 - TS2:

Si $\text{MICROSECOND}(\text{TS2}) \leq \text{MICROSECOND}(\text{TS1})$
entonces $\text{MICROSECOND}(\text{RESULT}) = \text{MICROSECOND}(\text{TS1}) - \text{MICROSECOND}(\text{TS2})$.

Si $\text{MICROSECOND}(\text{TS2}) > \text{MICROSECOND}(\text{TS1})$
entonces $\text{MICROSECOND}(\text{RESULT}) = 1000000 + \text{MICROSECOND}(\text{TS1}) - \text{MICROSECOND}(\text{TS2})$
y $\text{SECOND}(\text{TS2})$ se aumenta en 1.

Expresiones

La parte correspondiente a los segundos y minutos de la indicación de fecha y hora se resta tal como se especifica en las reglas para la resta de horas.

Si $\text{HOUR}(\text{TS2}) \leq \text{HOUR}(\text{TS1})$
entonces $\text{HOUR}(\text{RESULT}) = \text{HOUR}(\text{TS1}) - \text{HOUR}(\text{TS2})$.

Si $\text{HOUR}(\text{TS2}) > \text{HOUR}(\text{TS1})$
entonces $\text{HOUR}(\text{RESULT}) = 24 + \text{HOUR}(\text{TS1}) - \text{HOUR}(\text{TS2})$
y $\text{DAY}(\text{TS2})$ se aumenta en 1.

La parte correspondiente a la fecha de las indicaciones de la hora se resta tal como se especifica en las reglas para la resta de fechas.

Incremento y disminución de las indicaciones de fecha y hora: El resultado de sumar o restar una duración con una indicación de fecha y hora es también una indicación de fecha y hora. El cálculo con fechas y horas se realiza tal como se ha definido anteriormente, excepto que se acarrea un desbordamiento o subdesbordamiento a la parte de fecha del resultado, que debe estar dentro del rango de fechas válidas. El desbordamiento de microsegundos pasa a segundos.

Prioridad de operaciones

Las expresiones entre paréntesis y las expresiones de desreferencia se evalúan primero de izquierda a derecha.³⁰ Cuando del orden de evaluación no se especifica mediante paréntesis, los operadores de prefijo se aplican antes que la multiplicación y división, y la multiplicación y división se aplican antes que la suma y la resta. Los operadores de un mismo nivel de prioridad se aplican de izquierda a derecha.

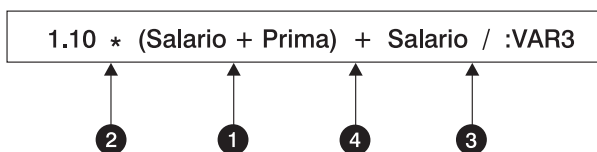
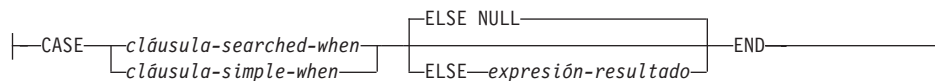


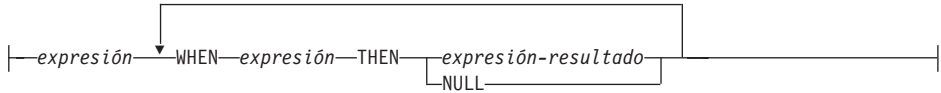
Figura 11. Prioridad de las operaciones

Expresiones CASE

expresión-case:



30. Tenga en cuenta que los paréntesis también se utilizan en sentencias de subselección, condiciones de búsqueda y funciones. Sin embargo, no deben utilizarse para agrupar arbitrariamente secciones dentro de sentencias de SQL.

cláusula-searched-when:**cláusula-simple-when:**

Las expresiones CASE permiten seleccionar una expresión en función de la evaluación de una o varias condiciones. En general, el valor de la expresión-case es el valor de la *expresión-resultado* que sigue a la primera (más a la izquierda) expresión case que se evalúa como cierta. Si ninguna se evalúa como cierta y está presente la palabra clave ELSE, el resultado es el valor de la *expresión-resultado* o NULL. Si ninguna se evalúa como cierta y no se utiliza la palabra clave ELSE, el resultado es NULL. Tenga presente que cuando una expresión CASE se evalúa como desconocida (debido a valores NULL), la expresión CASE no es cierta y por eso se trata igual que una expresión CASE que se evalúa como falsa.

Si la expresión CASE está en una cláusula VALUES, un predicado IN, una cláusula GROUP BY o en una cláusula ORDER BY, la *condición-búsqueda* de una cláusula-searched-when no puede ser un predicado cuantificado, un predicado IN que hace uso de una selección completa ni un predicado EXISTS (SQLSTATE 42625).

Cuando se utiliza la *cláusula-simple-when*, se comprueba si el valor de la *expresión* anterior a la primera palabra clave WHEN es igual al valor de la *expresión* posterior a la palabra clave WHEN. Por lo tanto, el tipo de datos de la *expresión* anterior a la primera palabra clave WHEN debe ser comparable a los tipos de datos de cada *expresión* posterior a la palabra o palabras clave WHEN. La *expresión* anterior a la primera palabra clave WHEN de una *cláusula-simple-when* no puede incluir ninguna función que sea una variante o que tenga una acción externa (SQLSTATE 42845).

Una *expresión-resultado* es una *expresión* que sigue a las palabras clave THEN o ELSE. Debe haber, como mínimo, una *expresión-resultado* en la expresión CASE (NULL no puede especificarse para cada case) (SQLSTATE 42625). Todas las *expresiones-resultado* deben tener tipos de datos compatibles (SQLSTATE

Expresiones

42804), donde los atributos del resultado se determinan basándose al apartado “Reglas para los tipos de datos del resultado” en la página 120.

Ejemplos:

- Si el primer carácter de un número de departamento corresponde a una división dentro de la organización, se puede utilizar una expresión CASE para listar el nombre completo de la división a la que pertenece cada empleado:

```
SELECT EMPNO, LASTNAME,  
       CASE SUBSTR(WORKDEPT,1,1)  
       WHEN 'A' THEN 'Administración'  
       WHEN 'B' THEN 'Recursos humanos'  
       WHEN 'C' THEN 'Contabilidad'  
       WHEN 'D' THEN 'Diseño'  
       WHEN 'E' THEN 'Operaciones'  
       END  
FROM EMPLOYEE;
```

- El número de años de formación académica se usa en la tabla EMPLOYEE para obtener el nivel de formación. Una expresión CASE se puede utilizar para agrupar estos datos y para mostrar el nivel de formación.

```
SELECT EMPNO, FIRSTNAME, MIDINIT, LASTNAME,  
       CASE  
       WHEN EDLEVEL < 15 THEN 'SECONDARY'  
       WHEN EDLEVEL < 19 THEN 'COLLEGE'  
       ELSE 'POST GRADUATE'  
       END  
FROM EMPLOYEE
```

- Otro ejemplo interesante del uso de una expresión CASE consiste en la protección de los errores que surjan de una división por 0. Por ejemplo, el siguiente código detecta los empleados que perciben más de un 25% de sus ingresos en comisiones, pero que su sueldo no se basa enteramente en comisiones.

```
SELECT EMPNO, WORKDEPT, SALARY+COMM FROM EMPLOYEE  
WHERE (CASE WHEN SALARY=0 THEN NULL  
        ELSE COMM/SALARY  
        END) > 0.25;
```

- Las siguientes expresiones CASE son iguales:

```
SELECT LASTNAME,  
       CASE  
       WHEN LASTNAME = 'Haas' THEN 'Presidente'  
       ...  
SELECT LASTNAME,  
       CASE LASTNAME  
       WHEN 'Haas' THEN 'Presidente'  
       ...
```

Existen dos funciones escalares, NULLIF y COALESCE, que sirven exclusivamente para manejar un subconjunto de la funcionalidad que una

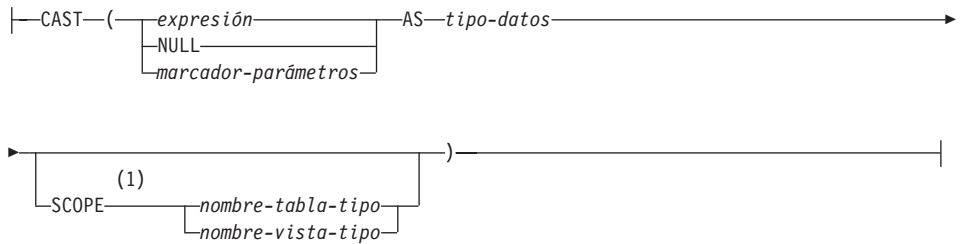
expresión CASE puede ofrecer. La Tabla 11 muestra la expresión equivalente al utilizar CASE o estas funciones.

Tabla 11. Expresiones CASE equivalentes

Expresión	Expresión equivalente
CASE WHEN e1=e2 THEN NULL ELSE e1 END	NULLIF(e1,e2)
CASE WHEN e1 IS NOT NULL THEN e1 ELSE e2 END	COALESCE(e1,e2)
CASE WHEN e1 IS NOT NULL THEN e1 ELSE COALESCE(e2,...,eN) END	COALESCE(e1,e2,...,eN)

Especificaciones CAST

especificación-cast:



Notas:

- 1 La cláusula SCOPE sólo se aplica al tipo de datos REF.

La especificación CAST devuelve el operando cast (el primer operando) convertido al tipo especificado por el *tipo de datos*.

expresión

Si el operando cast es una expresión (distinta del marcador de parámetros o NULL), el resultado es el valor del argumento convertido al *tipo de datos* de destino especificado.

Las especificaciones cast a las que se da soporte se muestran en la Tabla 6 en la página 104 donde la primera columna representa el tipo de datos del operando cast (tipo de datos fuente) y los tipos de datos en la parte superior representan el tipo de datos de destino de la especificación CAST. Si no se da soporte a la especificación cast, se producirá un error (SQLSTATE 42846).

Al convertir series de caracteres (que no sean CLOB) en una serie de caracteres de longitud diferente, se devuelve un aviso (SQLSTATE 01004) si se truncan otros caracteres que no sean los blancos de cola. Al convertir series de caracteres gráficas (que no sean DBCLOB) en una serie de

caracteres gráfica con una longitud diferente, se devuelve un aviso (SQLSTATE 01004) si se truncan otros caracteres que no sean los blancos de cola. Para los operandos BLOB, CLOB y DBCLOB de cast, el mensaje de aviso aparece si se trunca cualquier carácter.

NULL

Si el operando cast es la palabra clave NULL, el resultado es un valor nulo que tiene el *tipo de datos* especificado.

marcador-parámetros

Un marcador de parámetros (especificado como un signo de interrogación) se suele considerar como una expresión pero se documenta independientemente en este caso porque tiene un significado especial. Si el operando cast es un *marcador-parámetros*, el *tipo de datos* especificado se considera una promesa de que la sustitución se podrá asignar al tipo de datos especificado (utilizando la asignación de almacenamiento para series). Un marcador de parámetros como este se considera un *marcador de parámetros con tipo*. Los marcadores de parámetros con tipo se tratan como cualquier otro valor con tipo en lo referente a la resolución de funciones, a DESCRIBE de una lista de selección o a la asignación de columnas.

tipo de datos

Nombre de un tipo de datos existente. Si el nombre de tipo no está calificado, la vía de acceso de SQL se utiliza para realizar la resolución del tipo de datos. Un tipo de datos que tenga asociados atributos como, por ejemplo, la longitud o la precisión y escala debe incluir dichos atributos al especificar el *tipo de datos* (CHAR toma por omisión la longitud de 1 y DECIMAL toma por omisión una precisión de 5 y una escala de 0 si no se especifican). Las restricciones sobre los tipos de datos soportados se basan en el operando cast especificado.

- Para un operando cast que sea una *expresión*, vea “Conversión entre tipos de datos” en la página 101 para conocer los tipos de datos de destino a los que se da soporte según el tipo de datos del operando cast (tipo de datos fuente).
- Para un operando cast que sea la palabra clave NULL se puede utilizar cualquier tipo de datos existente.
- Para un operando cast que sea un marcador de parámetros, el tipo de datos de destino puede ser cualquier tipo de datos existente. Si el tipo de datos es un tipo diferenciado definido por el usuario, la aplicación que hace uso del marcador de parámetros utilizará el tipo de datos fuente del tipo diferenciado definido por el usuario. Si el tipo de datos es un tipo estructurado definido por el usuario, la aplicación que hace uso del marcador de parámetros utilizará el tipo de parámetro de entrada de la función de transformación TO de SQL para el tipo estructurado definido por el usuario.

SCOPE

Cuando el tipo de datos es un tipo de referencia, puede definirse un ámbito que identifique la tabla de destino o la vista de destino de la referencia.

nombre-tabla-tipo

El nombre de una tabla con tipo. Ya debe existir la tabla (SQLSTATE 42704). La conversión debe hacerse hacia el *tipo-datos* REF(S), donde S es el tipo de *nombre-tabla-tipo* (SQLSTATE 428DM).

nombre-vista-tipo

El nombre de una vista con tipo. La vista debe existir o tener el mismo nombre que la vista a crear que incluye la conversión del tipo de datos como parte de la definición de la vista (SQLSTATE 42704). La conversión debe hacerse hacia el *tipo-datos* REF(S), donde S es el tipo de *nombre-vista-tipo* (SQLSTATE 428DM).

Cuando se convierten datos numéricos en datos de caracteres, el tipo de datos resultante es una serie de caracteres de longitud fija (vea "CHAR" en la página 289). Cuando se convierten datos de caracteres en datos numéricos, el tipo de datos resultante depende del tipo de número especificado. Por ejemplo, si se convierte hacia un entero, pasará a ser un entero grande (vea "INTEGER" en la página 353).

Ejemplos:

- A una aplicación sólo le interesa la parte entera de SALARY (definido como decimal (9,2)) de la tabla EMPLOYEE. Se podría preparar la siguiente consulta, con el número de empleado y el valor del entero de SALARY.

```
SELECT EMPNO, CAST(SALARY AS INTEGER) FROM EMPLOYEE
```

- Supongamos que hay un tipo diferenciado denominado T_AGE que se define como SMALLINT y se utiliza para crear la columna AGE en la tabla PERSONNEL. Supongamos también que existe también un tipo diferenciado denominado R_YEAR que está definido en INTEGER y que se utiliza para crear la columna RETIRE_YEAR en la tabla PERSONNEL. Se podría preparar la siguiente sentencia de actualización.

```
UPDATE PERSONNEL SET RETIRE_YEAR =?  
WHERE AGE = CAST( ? AS T_AGE)
```

El primer parámetro es un marcador de parámetros no tipificado que tendría un tipo de datos de R_YEAR, si bien la aplicación utilizará un entero para este marcador de parámetros. Esto no necesita la especificación explícita de CAST porque se trata de una asignación.

El segundo marcador de parámetros es un marcador de parámetros con tipo que se convierte como un tipo diferenciado T_AGE. Esto cumple el requisito de que la comparación debe realizarse con tipos de datos

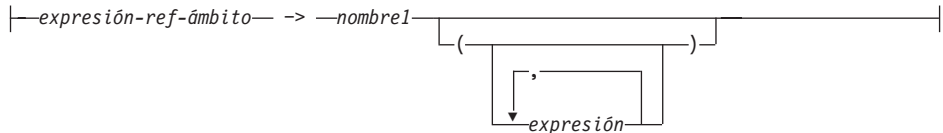
Expresiones

compatibles. La aplicación utilizará el tipo de datos fuente (que es SMALLINT) para procesarlo con este marcador de parámetros.

El proceso satisfactorio de esta sentencia supone que la vía de acceso de función incluye el nombre de esquema del esquema (o esquemas) donde están definidos los dos tipos diferenciados.

Operaciones de desreferencia

operación-desreferencia:



El ámbito de la expresión de referencia con ámbito es una tabla o vista llamada tabla o vista *destino*. La expresión de referencia con ámbito identifica una *fila destino*. La *fila destino* es la fila de la tabla o vista destino (o de una sus subtablas o subvistas) cuyo valor de la columna de identificador de objeto (OID) coincide con la expresión de referencia. Vea “CREATE TABLE” en la página 835 para obtener más información sobre las columnas de identificador de objeto. Se puede utilizar la operación de desreferencia para acceder a una columna de la fila destino, o para invocar un método, utilizando la fila destino como sujeto del método. El resultado de una operación de desreferencia puede siempre ser nulo. La operación de desreferencia tiene prioridad por encima de todos los otros operadores.

expresión-ref-ámbito

Una expresión que es un tipo de referencia que tiene un ámbito (SQLSTATE 428DT). Si la expresión es una variable del lenguaje principal, un marcador de parámetros u otro valor de tipo de referencia sin ámbito, se necesita una especificación CAST con una cláusula SCOPE para proporcionar un ámbito a la referencia.

nombre1

Especifica un identificador no calificado.

Si *nombre1* no va seguido por ningún paréntesis y *nombre1* coincida con el nombre de un atributo del tipo destino, el valor de la operación de desreferencia es el valor de la columna mencionada de la fila destino. En este caso, el tipo de datos de la columna (que puede contener nulos) determina el tipo del resultado de la operación de desreferencia. Si no existe ninguna fila destino cuyo identificador de objeto coincida con la expresión de referencia, el resultado de la operación de desreferencia es

nulo. Si la operación de desreferencia se utiliza en una lista de selección y no se incluye como parte de una expresión, *nombre1* pasa a ser el nombre de la columna resultante.

Si *nombre1* va seguido por un paréntesis o *nombre1* no coincida con el nombre de un atributo del tipo destino, la operación de desreferencia se trata como una invocación de método. El nombre del método invocado es *nombre1*. El sujeto del método es la fila destino, que se considera como una instancia de su tipo estructurado. Si no existe ninguna fila destino cuyo identificador de objeto coincida con la expresión de referencia, el sujeto del método es un valor nulo del tipo destino. Las expresiones entre paréntesis, si las hay, proporcionan los restantes parámetros de la invocación del método. El proceso normal se utiliza para la resolución de la invocación del método. El tipo resultante del método seleccionado (que puede contener nulos) determina el tipo resultante de la operación de desreferencia.

El ID de autorización de la sentencia que utiliza una operación de desreferencia debe tener el privilegio SELECT sobre la tabla de destino de la *expresión-ref-ámbito* (SQLSTATE 42501).

Una operación de desreferencia no puede nunca modificar valores de la base de datos. Si se utiliza una operación de desreferencia para invocar un método mutador, éste modifica una copia de la fila destino y devuelve la copia, dejando inalterada la base de datos.

Ejemplos:

- Suponga que existe una tabla EMPLOYEE que contiene una columna denominada DEPTREF, que es un tipo de referencia con ámbito para una tabla con tipo basada en un tipo que incluye el atributo DEPTNAME. Los valores de DEPTREF de la tabla EMPLOYEE deben corresponderse con los valores de la columna de OID de la tabla de destino de la columna DEPTREF.

```
SELECT EMPNO, DEPTREF->DEPTNAME
FROM EMPLOYEE
```

- Utilizando las mismas tablas que en el ejemplo anterior, utilice una operación de desreferencia para invocar un método llamado BUDGET, con la fila destino como parámetro sujeto y '1997' como parámetro adicional.

```
SELECT EMPNO, DEPTREF->BUDGET('1997')
AS DEPTBUDGET97
FROM EMPLOYEE
```

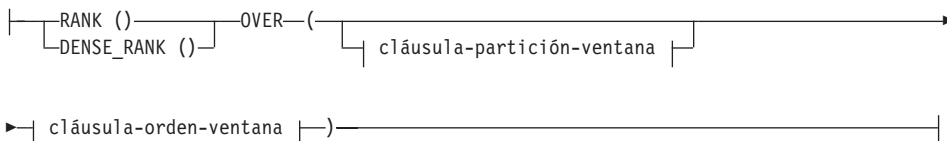
Funciones OLAP

función-OLAP:

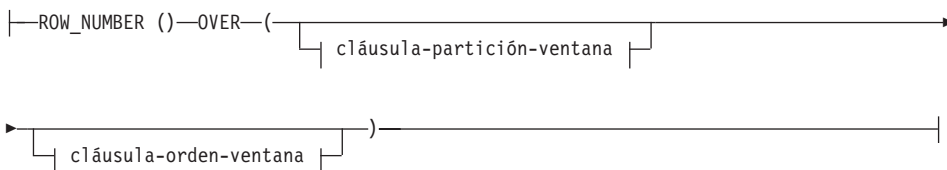
Expresiones



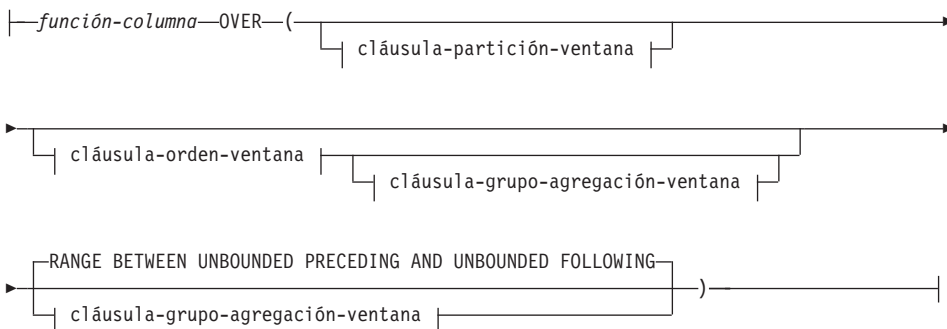
función-ordenación:



función-numeración:



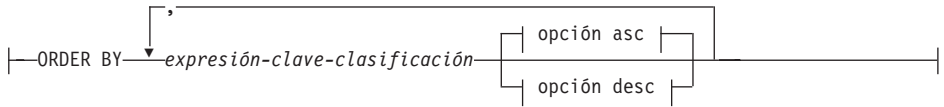
función-agregación:



cláusula-partición-ventana:



cláusula-orden-ventana:



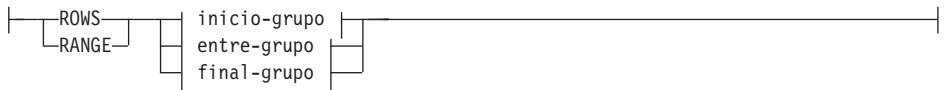
opción asc:



opción desc:



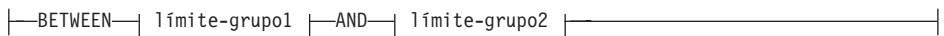
cláusula-grupo-agregación-ventana:



inicio-grupo:



entre-grupo:

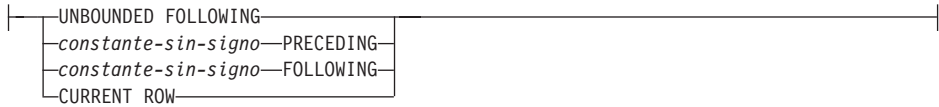


límite-grupo1:

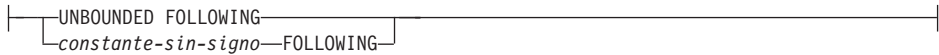


Expresiones

límite-grupo2:



final-grupo:



Las funciones OLAP (On-Line Analytical Processing) devuelven información sobre ordenación y numeración de filas y sobre funciones de columna existentes, así como un valor escalar en el resultado de una consulta. Se puede incluir una función OLAP en expresiones, en una lista de selección o en la cláusula ORDER BY de una sentencia SELECT (SQLSTATE 42903). Las funciones OLAP no se pueden utilizar como argumento de una función de columna (SQLSTATE 42607). La función OLAP se aplica a la tabla resultante de la subselección más interna donde reside la función OLAP.

Cuando se utiliza una función OLAP, se especifica una ventana que define las filas a las que se aplica la función, y en qué orden. Cuando la función OLAP se utiliza con una función de columna, las filas pertinentes se pueden definir con más detalle, con respecto a la fila actual, en forma de rango o indicando un número de filas que preceden y siguen a la fila actual. Por ejemplo, dentro de una división por meses, se puede calcular un valor promedio respecto a los tres meses anteriores.

La función de ordenación calcula la posición ordinal de una fila dentro de la ventana. Las filas que no son distintas con respecto a la ordenación dentro de sus ventanas tienen asignada la misma posición. Los resultados de la ordenación se pueden definir con o sin huecos en los números que resultan de valores duplicados.

Si se especifica RANK, la posición de una fila se define como 1 más el número de filas que preceden estrictamente a la fila. Por lo tanto, si dos o más filas no difieren con respecto a la ordenación, habrá uno o más huecos en la numeración jerárquica secuencial.

Si se especifica `DENSE_RANK`³¹, el rango de una fila se define como 1 más el número de filas que la preceden que son distintas respecto a la ordenación. Por tanto, no habrá huecos en la numeración jerárquica secuencial.

La función `ROW_NUMBER`³² calcula el número secuencial de la fila dentro de la ventana definida por la ordenación, empezando por 1 para la primera fila. Si la cláusula `ORDER BY` no está especificada en la ventana, los números de fila se asignan a las filas en un orden arbitrario, tal como son devueltas por la subselección (no de acuerdo con ninguna cláusula `ORDER BY` de la sentencia-select).

El tipo de datos del resultado de `RANK`, `DENSE_RANK` o `ROW_NUMBER` es `BIGINT`. El resultado no puede ser nulo.

PARTITION BY (*expresión-particionamiento,...*)

Define la partición que se utiliza para aplicar la función. Una *expresión-particionamiento* es una expresión utilizada para definir el particionamiento del conjunto resultante. Cada *nombre-columna* referenciado en una expresión-particionamiento debe identificar, sin ambigüedades, una columna del conjunto resultante de la sentencia de subselección donde reside la función OLAP (SQLSTATE 42702 ó 42703). La longitud de cada expresión-particionamiento no debe ser mayor que 255 bytes (SQLSTATE 42907). Una expresión-particionamiento no puede incluir una selección completa escalar (SQLSTATE 42822) ni ninguna función que no sea determinista o que tenga una acción externa (SQLSTATE 42845).

ORDER BY (*expresión-clave-clasificación,...*)

Define la ordenación de las filas dentro de una partición que determina el valor de la función OLAP o el significado de los valores de fila en la cláusula-grupo-agregación-ventana (no define la ordenación del conjunto resultante de la consulta). Una *expresión-clave-clasificación* es una expresión utilizada para definir la ordenación de las filas dentro de una partición de ventana. Cada nombre-columna referenciado en una expresión-clave-clasificación debe identificar, sin ambigüedades, una columna del conjunto resultante de la subselección donde reside la función OLAP (SQLSTATE 42702 ó 42703). La longitud de cada expresión-clave-clasificación no debe ser mayor que 255 bytes (SQLSTATE 42907). Una expresión-clave-clasificación no puede incluir una selección completa escalar (SQLSTATE 42822) ni ninguna función que no sea determinista o que tenga una acción externa (SQLSTATE 42845). Esta cláusula es necesaria para las funciones `RANK` y `DENSE_RANK` (SQLSTATE 42601).

31. `DENSE_RANK` y `DENSERANK` son sinónimos.

32. `ROW_NUMBER` y `ROWNUMBER` son sinónimos.

ASC

Utiliza los valores de la expresión-clave-clasificación en orden ascendente.

NULLS FIRST

La ordenación de ventana tiene en cuenta los valores nulos antes de los valores no nulos en el orden de clasificación.

NULLS LAST

La ordenación de ventana tiene en cuenta los valores nulos antes de los valores no nulos en el orden de clasificación.

DESC

Utiliza los valores de la expresión-clave-clasificación en orden descendente. Se considera que los valores nulos ocupan la primera posición en la ordenación.

cláusula-grupo-agregación-ventana

El grupo de agregación de una fila R es un conjunto de filas definidas en relación a R (en la ordenación de las filas de la partición de R). Esta cláusula especifica el grupo de agregación. Si no se especifica esta cláusula, el valor por omisión es el mismo que RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW, lo que proporciona un resultado de agregación acumulativo.

ROWS

Indica que el grupo de agregación se define mediante el conteo de filas.

RANGE

Indica que el grupo de agregación se define mediante un valor de desplazamiento con respecto a una clave de clasificación.

inicio-grupo

Especifica el punto de inicio del grupo de agregación. El final del grupo de agrupación es la fila actual. La cláusula inicio-grupo es equivalente a una cláusula entre-grupo en la forma "BETWEEN inicio-grupo AND CURRENT ROW".

entre-grupo

Especifica el inicio y final del grupo de agregación basándose en ROWS o RANGE.

final-grupo

Especifica el punto final del grupo de agregación. El inicio del grupo de agregación es la fila actual. La especificación de la cláusula final-grupo es equivalente a la de una cláusula entre-grupo del formato "BETWEEN CURRENT ROW AND final-grupo".

UNBOUNDED PRECEDING

Incluye la partición completa que precede a la fila actual. Esto se

puede especificar con ROWS o RANGE. También se puede especificar con varias expresiones-clave-clasificación en la cláusula-orden-ventana.

UNBOUNDED FOLLOWING

Incluye la partición completa que sigue a la fila actual. Esto se puede especificar con ROWS o RANGE. También se puede especificar con varias expresiones-clave-clasificación en la cláusula-orden-ventana.

CURRENT ROW

Especifica el inicio o el final del grupo de agregación basándose en la fila actual. Si se especifica ROWS, la fila actual es el límite del grupo de agregación. Si se especifica RANGE, el límite del grupo de agregación incluye el conjunto de filas con los mismos valores para las *expresiones-clave-clasificación* que la fila actual. Esta cláusula no se puede especificar en *límite-grupo2* si *límite-grupo1* especifica el *valor* FOLLOWING.

valor PRECEDING

Especifica el rango o número de filas que preceden a la fila actual. Si se especifica ROWS, *valor* es un entero positivo que indica un número de filas. Si se especifica RANGE, el tipo de datos de *valor* debe ser comparable con el tipo de la expresión-clave-clasificación de la cláusula-orden-ventana. Sólo puede haber una sola expresión-clave-clasificación y el tipo de datos de esa expresión debe permitir la operación de resta. Esta cláusula no se puede especificar en *límite-grupo2* si *límite-grupo1* es CURRENT ROW o *valor* FOLLOWING.

valor FOLLOWING

Especifica el rango o número de filas que siguen a la fila actual. Si se especifica ROWS, *valor* es un entero positivo que indica un número de filas. Si se especifica RANGE, el tipo de datos de *valor* debe ser comparable con el tipo de la expresión-clave-clasificación de la cláusula-orden-ventana. Sólo puede haber una sola expresión-clave-clasificación y el tipo de datos de esa expresión debe permitir la operación de suma.

Ejemplos:

- Este ejemplo muestra la ordenación de los empleados, dispuestos por apellidos, de acuerdo con un salario total (salario más prima) que sea mayor que \$30.000.

```
SELECT EMPNO, LASTNAME, FIRSTNAME, SALARY+BONUS AS TOTAL_SALARY,
       RANK() OVER (ORDER BY SALARY+BONUS DESC) AS RANK_SALARY
FROM EMPLOYEE WHERE SALARY+BONUS > 30000
ORDER BY LASTNAME
```

Observe que si el resultado debe estar ordenado de acuerdo con la escala de salarios, debe sustituir ORDER BY LASTNAME por:

Expresiones

```
ORDER BY RANK_SALARY
```

o

```
ORDER BY RANK() OVER (ORDER BY SALARY+BONUS DESC)
```

- Este ejemplo ordena los departamentos de acuerdo con su salario total medio.

```
SELECT WORKDEPT, AVG(SALARY+BONUS) AS AVG_TOTAL_SALARY,  
       RANK() OVER (ORDER BY AVG(SALARY+BONUS) DESC) AS RANK_AVG_SAL  
FROM EMPLOYEE  
  
GROUP BY WORKDEPT  
ORDER BY RANK_AVG_SAL
```

- Este ejemplo ordena los empleados de un departamento de acuerdo con su nivel de formación. Si varios empleados de un departamento tienen el mismo nivel, ello no debe suponer un aumento del nivel siguiente.

```
SELECT WORKDEPT, EMPNO, LASTNAME, FIRSTNME, EDLEVEL  
       DENSE_RANK() OVER  
       (PARTITION BY WORKDEPT ORDER BY EDLEVEL DESC) AS RANK_EDLEVEL  
FROM EMPLOYEE  
ORDER BY WORKDEPT, LASTNAME
```

- Este ejemplo proporciona números a las filas del resultado de una consulta.

```
SELECT ROW_NUMBER() OVER (ORDER BY WORKDEPT, LASTNAME) AS NUMBER,  
       LASTNAME, SALARY  
FROM EMPLOYEE  
ORDER BY WORKDEPT, LASTNAME
```

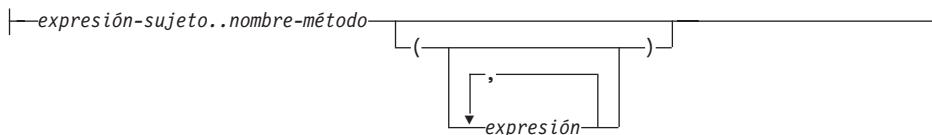
- Este ejemplo lista los cinco empleados con mayores ingresos.

```
SELECT EMPNO, LASTNAME, FIRSTNME, TOTAL_SALARY, RANK_SALARY  
FROM (SELECT EMPNO, LASTNAME, FIRSTNME, SALARY+BONUS AS TOTAL_SALARY,  
           RANK() OVER (ORDER BY SALARY+BONUS DESC) AS RANK_SALARY  
FROM EMPLOYEE) AS RANKED_EMPLOYEE  
WHERE RANK_SALARY < 6  
ORDER BY RANK_SALARY
```

Observe que primero se ha utilizado una expresión de tabla anidada para calcular el resultado, incluidos los niveles de ordenación, para poder utilizar el nivel en la cláusula WHERE. También se podría haber utilizado una expresión de tabla común.

Invocación de métodos

invocación-método:



El método observador y el método mutador, ambos generados por el sistema, así como los métodos definidos por el usuario se invocan utilizando el operador formado por dos puntos.

expresión-sujeto

Es una expresión con un tipo resultante estático que es un tipo estructurado definido por el usuario.

nombre-método

Es el nombre no calificado de un método. El tipo estático de *expresión-sujeto* o uno de sus supertipos debe incluir un método que tenga el nombre especificado.

(expresión,...)

Los argumentos de *nombre-método* se especifican entre paréntesis. Se pueden utilizar paréntesis vacíos para indicar que no existen argumentos. El *nombre-método* y los tipos de datos de las expresiones argumento especificadas se utilizan para obtener el método específico, basándose en el tipo estático de *expresión-sujeto* (vea "Resolución de métodos" en la página 170 para obtener más información).

El operador .. utilizado para invocar el método es un operador infijo que define una prioridad de operaciones de izquierda a derecha. Por ejemplo, las dos expresiones siguientes son equivalentes:

$$a..b..c + x..y..z$$

y

$$((a..b)..c) + ((x..y)..z)$$

Si un método no tiene ningún otro parámetro que no sea su sujeto, se puede invocar con o sin paréntesis. Por ejemplo, las dos expresiones siguientes son equivalentes:

$$\text{point1}..x$$

y

$$\text{point1}..x()$$

Los sujetos nulos de una invocación de método se manejan de este modo:

1. Si un método mutador generado por el sistema se invoca con un sujeto nulo, se produce un error (SQLSTATE 2202D)
2. Si cualquier método distinto de un método mutador generado por el sistema se invoca con un sujeto nulo, el método no se ejecuta y su resultado es nulo. Esta regla incluye los métodos definidos por el usuario con SELF AS RESULT.

Expresiones

Cuando se crea un objeto de base de datos (por ejemplo, un paquete, vista o desencadenante), se determina el método de ajuste óptimo que existe para cada invocación de método, utilizando las reglas especificadas en “Resolución de métodos” en la página 170.

Nota:

- Los métodos de los tipos definidos con WITH FUNCTION ACCESS también se pueden invocar utilizando la notación normal de funciones. La resolución de funciones considera como aceptables todas las funciones, así como los métodos con acceso a función. Sin embargo, las funciones no se pueden invocar utilizando la invocación de método. La resolución de métodos considera como aceptables todos los métodos, pero no las funciones. Si la resolución no proporciona una función o método apropiado, se produce un error (SQLSTATE 42884).

Ejemplos:

- Este ejemplo utiliza el operador .. para invocar un método llamado AREA. Se supone que existe una tabla llamada RINGS, con una columna CIRCLE_COL del tipo estructurado CIRCLE. Se supone también que el método AREA se ha definido previamente para el tipo CIRCLE como AREA() RETURNS DOUBLE.

```
SELECT CIRCLE_COL..AREA()  
FROM RINGS
```

Tratamiento de los subtipos

tratamiento-subtipo:

|—TREAT—(—expresión—AS—tipo-datos—)|

El *tratamiento-subtipo* se utiliza para convertir una expresión de tipo estructurado en uno de sus subtipos. El tipo estático de *expresión* debe ser un tipo estructurado definido por el usuario, y ese tipo debe ser el mismo que *tipo-datos* o que un subtipo de él. Si el nombre de tipo especificado en *tipo-datos* no está calificado, se utiliza la vía de acceso de SQL para resolver la referencia al tipo. El tipo estático del resultado de *tratamiento-subtipo* es *tipo-datos*, y el valor del *tratamiento-subtipo* es el valor de la expresión. Durante la ejecución, si el tipo dinámico de la expresión no es *tipo-datos* o un subtipo de *tipo-datos*, se produce un error (SQLSTATE 0D000).

Ejemplos:

- En este ejemplo, todas las instancias de objetos de la columna CIRCLE_COL están definidas con el tipo dinámico COLOREDCIRCLE para una aplicación. Se utiliza la consulta siguiente para invocar el método RGB para

tales objetos. Se supone que existe una tabla llamada RINGS, con una columna CIRCLE_COL del tipo estructurado CIRCLE. Se supone también que COLOREDCIRCLE es un subtipo de CIRCLE y que el método RGB se ha definido previamente para COLOREDCIRCLE como RGB() RETURNS DOUBLE.

```
SELECT TREAT (CIRCLE_COL AS COLOREDCIRCLE)..RGB()
FROM RINGS
```

Durante la ejecución, si hay instancias del tipo dinámico CIRCLE, se produce un error (SQLSTATE 0D000). Este error se puede evitar utilizando el predicado TYPE en una expresión CASE, del modo siguiente:

```
SELECT (CASE
WHEN CIRCLE_COL IS OF (COLOREDCIRCLE)
THEN TREAT (CIRCLE_COL AS COLOREDCIRCLE)..RGB()
ELSE NULL
END)
FROM RINGS
```

Vea “Predicado TYPE” en la página 230 para obtener más información.

Referencia de secuencia

referencia-secuencia:



expresión-nextval:



expresión-prevval:



NEXTVAL FOR *nombre-secuencia*

Una expresión NEXTVAL genera y devuelve el siguiente valor de la secuencia especificada por *nombre-secuencia*.

PREVVAL FOR *nombre-secuencia*

Una expresión PREVVAL devuelve el valor generado más recientemente de la secuencia especificada para una sentencia anterior del proceso de aplicación actual. Se puede hacer referencia a este valor repetidamente utilizando expresiones PREVVAL que especifican el nombre de la secuencia. Pueden existir múltiples instancias de las expresiones

Expresiones

PREVVAL especificando el mismo nombre de secuencia en una sola sentencia; todas ellas devuelven el mismo valor.

Una expresión PREVVAL sólo se puede utilizar si ya se ha hecho referencia a una expresión NEXTVAL que especifica el mismo nombre de secuencia en el proceso de aplicación actual, ya sea en la transacción actual ya sea en una transacción anterior (SQLSTATE 51035).

Nota:

- Se genera un valor nuevo para una secuencia cuando la expresión NEXTVAL especifica el nombre de dicha secuencia. Sin embargo, si existen múltiples instancias de una expresión NEXTVAL que especifican el mismo nombre de secuencia en una consulta, el contador para la secuencia se incrementa sólo una vez para cada fila del resultado y todas las instancias de NEXTVAL devuelven el mismo valor para una fila del resultado.
- Se puede utilizar el mismo número de secuencia como valor de clave de unicidad en dos tablas independientes haciendo referencia al número de secuencia con una expresión NEXTVAL para la primera fila (esto genera el valor de secuencia) y una expresión CURRVAL para las demás filas (la instancia de CURRVAL hace referencia al valor de secuencia generado más recientemente en la sesión actual), como se muestra a continuación:

```
INSERT INTO order(orderno, cutno)
VALUES (NEXTVAL FOR order_seq, 123456);
```

```
INSERT INTO line_item (orderno, partno, quantity)
VALUES (PREVVAL FOR order_seq, 987654, 1);
```

- Las expresiones NEXTVAL y PREVVAL pueden especificarse en los lugares siguientes:
 - sentencia-select o sentencia SELECT INTO (en la cláusula-select, a condición de que la sentencia no contenga una palabra clave DISTINCT, una cláusula GROUP BY, una cláusula ORDER BY, una palabra clave UNION, una palabra clave INTERSECT o una palabra clave EXCEPT)
 - sentencia INSERT (en una cláusula VALUES)
 - sentencia INSERT (en la cláusula-select de la selección completa (fullselect))
 - sentencia UPDATE (en la cláusula SET (una sentencia UPDATE buscada o colocada), excepto que no se puede especificar NEXTVAL en la cláusula-select de la selección completa de una expresión de la cláusula SET)
 - sentencia VALUES INTO (en la cláusula-select de la selección completa (fullselect) de una expresión)

- sentencia CREATE PROCEDURE (en el cuerpo-rutina de un procedimiento SQL)
- sentencia CREATE TRIGGER en la acción-desencadenada (se puede especificar una expresión NEXTVAL, pero no se puede especificar una expresión PREVVAL)
- No se pueden especificar expresiones NEXTVAL y PREVVAL (SQLSTATE 428F9) en los lugares siguientes:
 - las condiciones de unión de una unión externa completa
 - el valor DEFAULT de una columna en una sentencia CREATE o ALTER TABLE
 - la definición de columna generada en una sentencia CREATE o ALTER TABLE
 - la definición de tabla de resumen de una sentencia CREATE TABLE o ALTER TABLE
 - la condición de una restricción CHECK
 - la sentencia CREATE TRIGGER (se puede especificar una expresión NEXTVAL, pero no se puede especificar una expresión PREVVAL)
 - la sentencia CREATE VIEW
 - la sentencia CREATE METHOD
 - la sentencia CREATE FUNCTION
- Además, no se puede especificar una expresión NEXTVAL (SQLSTATE 428F9) en los lugares siguientes:
 - la expresión CASE
 - la lista de parámetros de una función agregada
 - la subconsulta en un contexto distinto de los explícitamente permitidos mencionados anteriormente
 - la sentencia SELECT para la que la SELECT externa contiene un operador DISTINCT
 - la condición de unión de una unión
 - la sentencia SELECT para la que la SELECT externa contiene una cláusula GROUP BY
 - la sentencia SELECT para la que la SELECT externa está combinada con otra sentencia SELECT utilizando el operador establecido UNION, INTERSECT o EXCEPT
 - la expresión de tabla anidada
 - la lista de parámetros de una función de tabla
 - la cláusula WHERE de la sentencia SELECT más externa o una sentencia DELETE o UPDATE
 - la cláusula ORDER BY de la sentencia SELECT más externa

- la cláusula-select de la selección completa (fullselect) de una expresión, en la cláusula SET de una sentencia UPDATE
- la sentencia IF, WHILE, DO ... UNTIL o CASE de una rutina SQL
- Cuando se genera un valor para una secuencia, se consume dicho valor y, la siguiente vez que se solicita un valor, se genera un valor nuevo. Esto es válido incluso cuando la sentencia que contiene la expresión NEXTVAL falla o se retrotrae.

Si una sentencia INSERT incluye una expresión NEXTVAL en la lista VALUES para la columna y si se produce un error en algún punto durante la ejecución de INSERT (puede ser un problema al generar el siguiente valor de secuencia o un problema con el valor de otra columna), se produce una anomalía de inserción (SQLSTATE 23505) y se considera que el valor generado para la secuencia se ha consumido. En algunos casos, al volver a emitir la misma sentencia INSERT se puede obtener un resultado satisfactorio.

Por ejemplo, considere un error que es el resultado de la existencia de un índice de unicidad para la columna para la que se ha utilizado NEXTVAL y el valor de secuencia generado ya existe en el índice. Es posible que el siguiente valor generado para la secuencia sea un valor que no existe en el índice y, por consiguiente, el INSERT subsiguiente dará un resultado satisfactorio.

- Si al generar un valor para una secuencia, se excede el valor máximo para la secuencia (o el valor mínimo para una secuencia descendente) y no se permiten ciclos, se producirá un error (SQLSTATE 23522). En este caso, el usuario puede modificar (ALTER) la secuencia para ampliar el rango de valores aceptables, habilitar ciclos para la secuencia o eliminar (DROP) la secuencia y crear (CREATE) una nueva con un tipo de datos diferente que tenga un mayor rango de valores.

Por ejemplo, una secuencia puede haberse definido con un tipo de datos de SMALLINT y, finalmente, la secuencia se queda sin valores asignables. Elimine (DROP) y vuelva a crear la secuencia con la nueva definición para volver a definir la secuencia como INTEGER.

- Una referencia a una expresión NEXTVAL en la sentencia de selección (select) de un cursor hace referencia a un valor que se genera para una fila de la tabla resultante. Se genera un valor de secuencia para una expresión NEXTVAL para cada fila que se busca desde la base de datos. Si se realiza el bloqueo en el cliente, puede que los valores se hayan generado en el servidor antes del proceso de la sentencia FETCH. Esto puede producirse cuando existe bloqueo de las filas de la tabla resultante. Si la aplicación cliente no busca (FETCH) explícitamente todas las filas que la base de datos ha

materializado, la aplicación no verá los resultados de todos los valores de secuencia generados (para las filas materializadas que no se ha devuelto).

- Una referencia a una expresión PREVVAL de la sentencia de selección (select) de un cursor hace referencia a un valor que se ha generado para la secuencia especificada antes de la apertura del cursor. Sin embargo, el cierre del cursor puede afectar a los valores devueltos por PREVVAL para la secuencia especificada en las sentencias futuras o incluso para la misma sentencia en el caso de que se vuelva a abrir el cursor. Esto sucederá cuando la sentencia de selección del cursor incluya una referencia a NEXTVAL para el mismo nombre de secuencia.

Ejemplos:

En estos ejemplos se supone que existe una tabla llamada "order" y que se crea una secuencia llamada "order_seq" del modo siguiente:

```
CREATE SEQUENCE order_seq
  START WITH 1
  INCREMENT BY 1
  NO MAXVALUE
  NO CYCLE
  CACHE 24
```

- Ejemplos de cómo generar un número de secuencia "order_seq" con una expresión NEXTVAL para la secuencia creada anteriormente:

```
INSERT INTO order(orderno, custno)
  VALUES (NEXTVAL FOR order_seq, 123456);
```

O

```
UPDATE order
  SET orderno = NEXTVAL FOR order_seq
  WHERE custno = 123456;
```

O

```
VALUES NEXTVAL FOR order_seq INTO :hv_seq;
```

Predicados

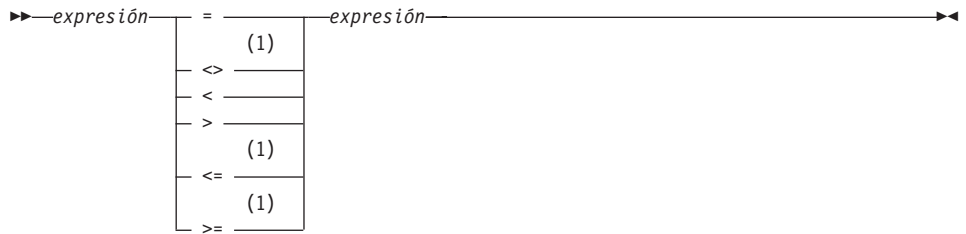
Un *predicado* especifica una condición que es cierta, falsa o desconocida acerca de una fila o un grupo determinado.

Las siguientes reglas se aplican a todos los tipos de predicados:

- Todos los valores especificados en un predicado debe ser compatibles.
- Una expresión que se utiliza en un predicado Básico, Cuantificado, IN o BETWEEN no debe dar como resultado una serie de caracteres con un atributo de longitud superior a 4.000, una serie de caracteres gráficos con un atributo de longitud superior a 2.000 ni una serie LOB de cualquier tamaño.
- El valor de una variable de lenguaje principal puede ser nulo (es decir, la variable puede tener una variable indicadora negativa).
- La conversión de la página de códigos de los operandos de los predicados que incluyen dos o más operandos, a excepción de LIKE, se realiza según el apartado “Reglas para las conversiones de series” en la página 124
- La utilización de un valor DATALINK se limita al predicado NULL.
- La utilización de un valor de tipo estructurado está limitado al predicado NULL y al predicado TYPE.

La selección completa (fullselect) es una forma de sentencia SELECT; está descrita en el “Capítulo 5. Consultas” en la página 471. Una selección completa utilizada en un predicado también se llama *subconsulta*.

Predicado básico



Notas:

- 1 También se da soporte a otros operadores de comparación ³³.

Un *predicado básico* compara dos valores.

Si el valor de cualquier operando es nulo, el resultado del predicado será desconocido. De lo contrario el resultado es verdadero o falso.

Para valores x e y :

Predicado	Es verdadero si y sólo si...
$x = y$	x es igual a y
$x \neq y$	x es diferente de y
$x < y$	x es menor que y
$x > y$	x es mayor que y
$x \geq y$	x es mayor o igual que y
$x \leq y$	x es menor o igual que y

Ejemplos:

```

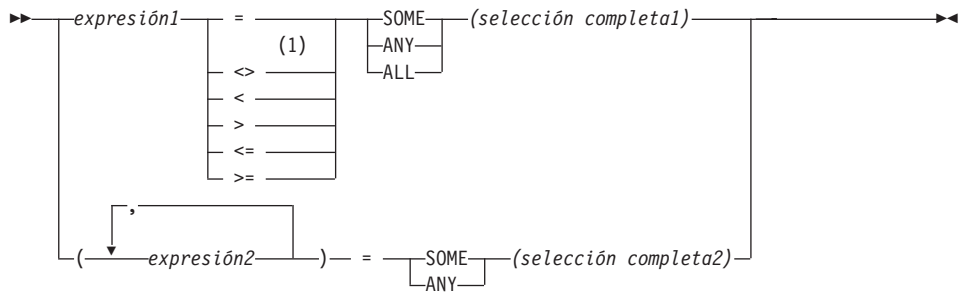
EMPNO='528671'
SALARY < 20000
PRSTAFF <> :VAR1
SALARY > (SELECT AVG(SALARY) FROM EMPLOYEE)
    
```

33. También se da soporte a los formatos siguientes de operadores de comparación en predicados básicos y cuantificados; $\hat{=}$, $\hat{<}$, $\hat{>}$, $\hat{!} =$, $\hat{!} <$ y $\hat{!} >$. Además, en las páginas de códigos 437, 819 y 850, se da soporte a los formatos $\neg =$, $\neg <$ y $\neg >$.

Estos formatos específicos de producto de los operadores de comparación solamente pretenden dar soporte al SQL existente que emplea estos operadores y no se recomienda utilizarlos al escribir sentencias de SQL nuevas.

Predicado cuantificado

Predicado cuantificado



Notas:

- 1 También se da soporte a otros operadores de comparación ³³

Un *predicado cuantificado* compara un valor o valores con un grupo de valores.

La selección completa debe identificar un número de columnas que sea el mismo que el número de expresiones especificadas a la izquierda del operador del predicado (SQLSTATE 428C4). La selección completa puede devolver cualquier número de filas.

Cuando se especifica ALL:

- El resultado del predicado es verdadero si la selección completa no devuelve ningún valor o si la relación especificada es verdadera para cada valor que devuelva la selección completa.
- El resultado es falso si la relación especificada es falsa para un valor como mínimo que devuelve la selección completa.
- El resultado es desconocido si la relación especificada no es falsa para ninguno de los valores que devuelve la selección completa y una comparación como mínimo es desconocida debido a un valor nulo.

Cuando se especifica SOME o ANY:

- El resultado del predicado es verdadero si la relación especificada es verdadera para cada valor de una fila como mínimo que devuelve la selección completa.
- El resultado es falso si la selección completa no devuelve ninguna fila o si la relación especificada es falsa para como mínimo un valor de cada fila que devuelve la selección completa.
- El resultado es desconocido si la relación especificada no es verdadera para cualquiera de las filas y, como mínimo, una comparación es desconocida debido a un valor nulo.

Ejemplos: Utilice las tablas siguientes al hacer referencia a los ejemplos siguientes.

COLA	COLB
1	12
2	12
3	13
4	14
-	-

COLX	COLY
2	22
3	23

Figura 12.

Ejemplo 1

```
SELECT COLA FROM TBLAB
WHERE COLA = ANY(SELECT COLX FROM TBLXY)
```

Da como resultado 2,3. La subselección devuelve (2,3). COLA en las filas 2 y 3 es igual al menos a uno de estos valores.

Ejemplo 2

```
SELECT COLA FROM TBLAB
WHERE COLA > ANY(SELECT COLX FROM TBLXY)
```

Da como resultado 3,4. La subselección devuelve (2,3). COLA en las filas 3 y 4 es mayor que al menos uno de estos valores.

Ejemplo 3

```
SELECT COLA FROM TBLAB
WHERE COLA > ALL(SELECT COLX FROM TBLXY)
```

Da como resultado 4. La subselección devuelve (2,3). COLA en la fila 4 es el único que es mayor que estos dos valores.

Ejemplo 4

```
SELECT COLA FROM TBLAB
WHERE COLA > ALL(SELECT COLX FROM TBLXY
WHERE COLX<0)
```

Da como resultado 1,2,3,4, nulo. La subselección no devuelve ningún valor. Por lo tanto, el predicado es verdadero para todas las filas de TBLAB.

Ejemplo 5

```
SELECT * FROM TBLAB
WHERE (COLA, COLB+10) = SOME (SELECT COLX, COLY FROM TBLXY)
```

Predicado cuantificado

La subselección devuelve todas las entradas de TBLXY. El predicado es verdadero para la subselección, por lo tanto el resultado es el siguiente:

COLA	COLB
2	12
3	13

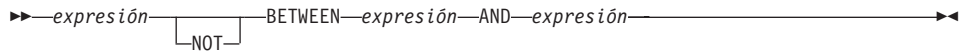
Ejemplo 6

```
SELECT * FROM TBLAB  
WHERE (COLA, COLB) = ANY (SELECT COLX, COLY-10 FROM TBLXY)
```

La subselección devuelve COLX y COLY-10 de TBLXY. El predicado es verdadero para la subselección, por lo tanto el resultado es el siguiente:

COLA	COLB
2	12
3	13

Predicado BETWEEN



El predicado BETWEEN compara un valor con un rango de valores.

El predicado BETWEEN:

`valor1 BETWEEN valor2 AND valor3`

es equivalente a la condición de búsqueda:

`valor1 >= valor2 AND valor1 <= valor3`

El predicado BETWEEN:

`valor1 NOT BETWEEN valor2 AND valor3`

es equivalente a la condición de búsqueda:

`NOT(valor1 BETWEEN valor2 AND valor3)`; es decir,
`valor1 < valor2 OR valor1 > valor3`.

Los valores de las expresiones del predicado BETWEEN pueden tener páginas de códigos diferentes. Los operandos se convierten como si se especificaran las condiciones de búsqueda equivalentes que se acaban de mencionar.

El primer operando (expresión) no puede incluir ninguna función que sea variante o que tenga una acción externa (SQLSTATE 426804).

En una mezcla de valores de indicación de fecha y hora y representaciones de serie de caracteres, todos los valores se convierten al tipo de datos del operando de fecha y hora.

Ejemplos:

Ejemplo 1

`EMPLOYEE.SALARY BETWEEN 20000 AND 40000`

Devuelve todos los salarios comprendidos entre 20.000 y 40.000 dólares.

Ejemplo 2

`SALARY NOT BETWEEN 20000 + :HV1 AND 40000`

Suponiendo que :HV1 es 5000, da como resultado todos los salarios que son inferiores a 25.000 dólares y superiores a 40.000.

Predicado BETWEEN

Ejemplo 3

Suponiendo lo siguiente:

Tabla 12.

Expresiones	Tipo	Página de códigos
HV_1	variable del lenguaje principal	437
HV_2	variable del lenguaje principal	437
Col_1	columna	850

Cuando se evalúa el predicado:

```
:HV_1 BETWEEN :HV_2 AND COL_1
```

Se interpretará como:

```
:HV_1 >= :HV_2  
AND :HV_1 <= COL_1
```

La primera ocurrencia de :HV_1 permanecerá en la página de códigos de la aplicación ya que se compara con :HV_2 que también permanece en la página de códigos de la aplicación. La segunda ocurrencia de :HV_1 se convertirá a la página de códigos de la base de datos ya que se compara con un valor de columna.

Predicado EXISTS

►►—EXISTS—(*selección completa*)—◄◄

El predicado EXISTS comprueba la existencia de ciertas filas.

La selección completa puede especificar cualquier número de columnas y

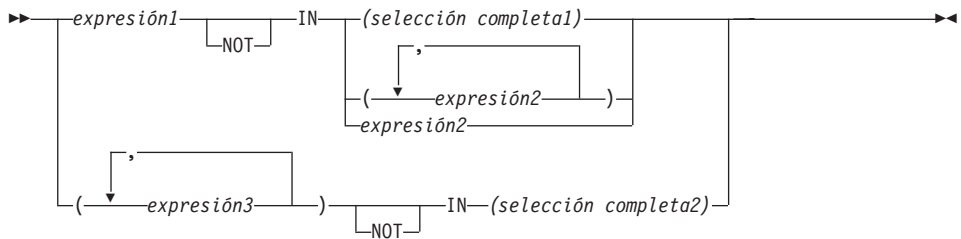
- El resultado es verdadero sólo si el número de filas especificadas mediante la selección completa no es cero.
- El resultado es falso sólo si el número de filas especificadas es cero
- El resultado no puede ser desconocido.

Ejemplo:

```
EXISTS (SELECT * FROM TEMPL WHERE SALARY < 10000)
```

Predicado IN

Predicado IN



El predicado IN compara un valor o valores con un conjunto de valores.

La selección completa debe identificar un número de columnas que sea el mismo que el número de expresiones especificadas a la izquierda de la palabra clave IN (SQLSTATE 428C4). La selección completa puede devolver cualquier número de filas.

- Un predicado IN del formato:

expresión **IN** expresión

es equivalente a un predicado básico del formato:

expresión = expresión

- Un predicado IN del formato:

expresión **IN** (selección completa)

es equivalente a un predicado cuantificado del formato:

expresión = **ANY** (selección completa)

- Un predicado IN del formato:

expresión **NOT IN** (selección completa)

es equivalente a un predicado cuantificado del formato:

expresión <> **ALL** (selección completa)

- Un predicado IN del formato:

expresión **IN** (expresióna, expresiónb, ..., expresiónk)

es equivalente a:

expresión = **ANY** (selección completa)

donde selección completa en el formato de la cláusula-values es:

VALUES (expresiónA), (expresiónB), ..., (expresiónK)

- Un predicado IN del formato:

(expresiónA, expresiónB,...,
expresiónK) IN (selección completa)

es equivalente a un predicado cuantificado del formato:

(expresiónA, expresiónB,..., expresiónK) = ANY (selección completa)

Los valores para *expresión1* y *expresión2* o la columna de *selección completa1* del predicado IN deben ser compatibles. Cada valor de *expresión3* y su columna correspondiente de *selección completa2* del predicado IN deben ser compatibles. Puede utilizarse el apartado “Reglas para los tipos de datos del resultado” en la página 120 para determinar los atributos del resultado utilizados en la comparación.

Los valores para las expresiones del predicado IN (incluyendo las columnas correspondientes de una selección completa) pueden tener páginas de códigos diferentes. Si se precisa realizar una conversión, la página de códigos se determina aplicando el apartado “Reglas para las conversiones de series” en la página 124 a la lista IN primero y, posteriormente, al predicado que utiliza la página de códigos derivada para la lista IN como segundo operando.

Ejemplos:

Ejemplo 1: Lo siguiente es verdadero si el valor de la fila bajo evaluación de la columna DEPTNO contiene D01, B01 o C01:

DEPTNO IN ('D01', 'B01', 'C01')

Ejemplo 2: Lo siguiente se considera verdadero sólo si EMPNO (número de empleado) a la izquierda coincide con EMPNO de un empleado del departamento E11:

EMPNO IN (SELECT EMPNO FROM EMPLOYEE WHERE WORKDEPT = 'E11')

Ejemplo 3: Dada la siguiente información, este ejemplo se considera verdadero si el valor específico de la fila de la columna COL_1 coincide con cualquier valor de la lista:

Tabla 13. Ejemplo de predicado IN

Expresiones	Tipo	Página de códigos
COL_1	columna variable del lenguaje	850
HV_2	principal variable del lenguaje	437
HV_3	principal	437
CON_1	constante	850

Cuando se evalúa el predicado:

Predicado IN

```
COL_1 IN (:HV_2, :HV_3, CON_4)
```

Las dos variables de lenguaje principal se convertirán a la página de códigos 850 tal como se indica en el apartado “Reglas para las conversiones de series” en la página 124.

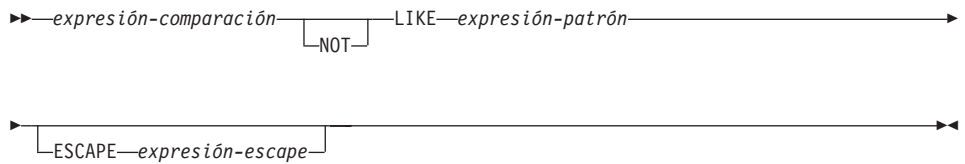
Ejemplo 4: Lo siguiente se considera verdadero si el año especificado en EMENDATE (la fecha en que ha finalizado la actividad de un empleado en un proyecto) coincide con cualquiera de los valores especificados en la lista (el año actual o los dos años anteriores):

```
YEAR(EMENDATE) IN (YEAR(CURRENT DATE),  
                  YEAR(CURRENT DATE - 1 YEAR),  
                  YEAR(CURRENT DATE - 2 YEARS))
```

Ejemplo 5: Lo siguiente se considera verdadero si tanto ID como DEPT del lado izquierdo coinciden con MANAGER y DEPTNUMB respectivamente para cualquier fila de la tabla ORG.

```
(ID, DEPT) IN (SELECT MANAGER, DEPTNUMB FROM ORG)
```

Predicado LIKE



El predicado LIKE busca series que sigan un patrón determinado. El patrón se especifica mediante una serie de caracteres en la que los signos de subrayado y de tanto por ciento tienen significados especiales. Los blancos de cola de un patrón también forman parte del patrón.

Si el valor de cualquier argumento es nulo, el resultado del predicado LIKE es desconocido.

Los valores de *expresión-comparación*, *expresión-patrón* y *expresión-escape* son expresiones de serie compatibles. Existen diferencias muy pequeñas en los tipos de expresiones de serie a las que se da soporte para cada uno de estos argumentos. Los tipos válidos de expresiones se listan en la descripción de cada argumento.

Ninguna de las expresiones puede dar como resultado un tipo diferenciado. No obstante, puede ser una función que convierte un tipo diferenciado a su tipo de fuente.

expresión-comparación

Expresión que especifica la serie que se debe examinar para ver si cumple determinados patrones de caracteres.

La expresión puede especificarse mediante:

- una constante
- un registro especial
- una variable del lenguaje principal (incluyendo una variable localizadora o una variable de referencia a archivos)
- una función escalar
- un localizador de gran objeto
- un nombre de columna
- una expresión que concatene cualquiera de los elementos anteriores

expresión-patrón

Expresión que especifica la serie que debe coincidir.

La expresión puede especificarse mediante:

Predicado LIKE

- una constante
- un registro especial
- una variable del lenguaje principal
- una función escalar cuyos operandos sean cualquiera de los elementos anteriores
- una expresión que concatene cualquiera de los elementos anteriores

teniendo en cuenta las siguientes restricciones:

- Ningún elemento de la expresión puede ser de tipo LONG VARCHAR, CLOB, LONG VARGRAPHIC o DBCLOB. Además, no puede ser una variable de referencia a archivos BLOB.
- La longitud real de *expresión-patrón* no puede ser mayor que 32.672 bytes.

Una **descripción simple** de la utilización del patrón LIKE es que el patrón se utiliza para especificar el criterio de conformidad para los valores de la *expresión-comparación* donde:

- El carácter de subrayado () representa cualquier carácter.
- El signo de porcentaje (%) representa una serie de ninguno o más caracteres.
- Cualquier otro carácter se representa a sí mismo.

Si la *expresión-patrón* necesita incluir el signo de subrayado o de porcentaje, se utiliza la *expresión-escape* para anteponer un carácter al carácter de subrayado o de porcentaje en el patrón.

A continuación encontrará una **descripción exacta** de la utilización del patrón LIKE. Observe que esta descripción no considera el uso de la *expresión-escape*; su uso se describe más adelante.

- Supongamos que m indique el valor de *expresión-comparación* y que p indique el valor de *expresión-patrón*. La serie p se interpreta como una secuencia del número mínimo de especificadores de subserie de modo que cada carácter de p forma parte exacta de un especificador de subserie. Un especificador de subserie es un signo de subrayado, un signo de tanto por ciento o bien una secuencia de caracteres no vacía que no sea un signo de subrayado ni de tanto por ciento.

El resultado del predicado es desconocido si m o p es el valor nulo. De lo contrario, el resultado es verdadero o falso. El resultado es verdadero si m y p son dos series vacías o existe una partición de m en subseries de modo que:

- Una subserie de m es una secuencia de cero o más caracteres continuos y cada carácter de m forma parte exacta de una subserie.

- Si el especificador de subserie n es un subrayado, la subserie n de m es cualquier carácter.
- Si el especificador de subserie n es un signo de tanto por ciento, la subserie n de m es cualquier secuencia de cero o más caracteres.
- Si el especificador de subserie n no es un subrayado ni un signo de tanto por ciento, la subserie n de m es igual al especificador de subserie y tiene la misma longitud que el especificador de subserie.
- El número de subseries de m es igual al número de especificadores de subserie.

Por consiguiente, si p es una serie vacía y m no es una serie vacía, el resultado es falso. Del mismo modo, si m es una serie vacía y p no es una serie vacía (excepto en el caso de una serie que sólo contenga signos de porcentaje), el resultado es falso.

El predicado m NOT LIKE p es equivalente a la condición de búsqueda NOT (m LIKE p).

Cuando se especifica la *expresión-escape*, la *expresión-patrón* no debe contener el carácter de escape identificado por la *expresión-escape* excepto cuando viene seguido inmediatamente por el carácter de escape, el carácter de subrayado o el carácter de tanto por ciento (SQLSTATE 22025).

Si la *expresión-comparación* es una serie de caracteres de una base de datos MBCS, puede contener **datos mixtos**. En este caso, el patrón puede estar compuesto por caracteres SBCS y MBCS. Los caracteres especiales del patrón se interpretan de la forma siguiente:

- Un signo de subrayado SBCS representa un carácter SBCS.
- Un signo de subrayado DBCS representa un carácter MBCS.
- Un signo de tanto por ciento (SBCS o DBCS) representa una serie de cero o más caracteres SBCS o MBCS.

expresión-escape

Este argumento opcional es una expresión que especifica un carácter que modifica el significado especial de los caracteres de subrayado (`_`) y de porcentaje (`%`) en la *expresión-patrón*. De este modo, el predicado LIKE permite comparar valores que contengan los caracteres de porcentaje y subrayado efectivos.

La expresión puede especificarse mediante:

- una constante
- un registro especial
- una variable del lenguaje principal

Predicado LIKE

- una función escalar cuyos operandos sean cualquiera de los elementos anteriores
- una expresión que concatene cualquiera de los elementos anteriores

teniendo en cuenta las siguientes restricciones:

- Ningún elemento de la expresión puede ser de tipo LONG VARCHAR, CLOB, LONG VARGRAPHIC o DBCLOB. Además, no puede ser una variable de referencia a archivos BLOB.
- El resultado de la expresión debe ser un carácter SBCS o DBCS o una serie binaria que contenga exactamente 1 byte (SQLSTATE 22019).

Cuando los caracteres de escape están presentes en la serie patrón, tanto un signo de carácter de subrayado, un signo de tanto por ciento como un carácter de escape pueden representar una ocurrencia literal por sí mismos. Esto es verdadero si el carácter viene precedido por un número impar de caracteres de escape sucesivos. De lo contrario, no es verdadero.

En un patrón, una secuencia de caracteres de escape sucesivos se trata de la siguiente manera:

- Supongamos que S es una secuencia y que no forma parte de una secuencia más grande de caracteres de escape sucesivos. Supongamos asimismo que S contiene en total n caracteres. Las reglas que rigen a S dependen del valor de n:
 - Si n es impar, S debe ir seguido por un carácter de subrayado o bien por un signo de tanto por ciento (SQLSTATE 22025). S y el carácter que le sigue representan $(n-1)/2$ ocurrencias literales del carácter de escape seguido por una ocurrencia literal del signo de subrayado o de tanto por ciento.
 - Si n es par, S representa $n/2$ ocurrencias literales del carácter de escape. A diferencia de cuando n es impar, S podría finalizar el patrón. Si no finaliza el patrón, puede ir seguido por cualquier carácter (a excepción, obviamente, de un carácter de escape, ya que vulneraría la suposición de que S no forma parte de una secuencia más grande de caracteres de escape sucesivos). Si S va seguido por un signo de subrayado o de tanto por ciento, ese carácter tiene su significado especial.

A continuación se ilustra el efecto de ocurrencias sucesivas del carácter de escape (que, en este caso, es la barra inclinada invertida (\)).

Serie patrón	Patrón real
\%	Un signo de tanto por ciento
\\%	Una barra invertida seguida por un cero o varios caracteres arbitrarios

\\% Una barra invertida seguida por un signo de tanto por ciento

La página de códigos utilizada en la comparación se basa en la página de códigos del valor de *expresión-comparación*.

- El valor de la *expresión-comparación* no se convierte nunca.
- Si la página de códigos de la *expresión-patrón* es diferente de la página de códigos de la *expresión-comparación*, el valor de la *expresión-patrón* se convierte a la página de códigos de la *expresión-comparación*, a menos que algún operando se haya definido como FOR BIT DATA (en cuyo caso no se efectúa la conversión).
- Si la página de códigos de la *expresión-escape* es diferente de la página de códigos de la *expresión-comparación*, el valor de la *expresión-escape* se convierte a la página de códigos de la *expresión-comparación*, a menos que algún operando se defina como FOR BIT DATA (en cuyo caso no se efectúa la conversión).

Notas

Si el patrón especificado en un predicado LIKE es un marcador de parámetro y se utiliza una variable del lenguaje principal de caracteres de longitud fija para sustituir el marcador de parámetro, el valor especificado para la variable del lenguaje principal debe tener la longitud correcta. Si no se especifica la longitud correcta, la selección no devolverá los resultados esperados.

Por ejemplo, si la variable del lenguaje principal está definida como CHAR(10) y se asigna el valor WYSE% a dicha variable del lenguaje principal, ésta se rellena con espacios en blanco en la asignación. El patrón utilizado es 'WYSE% '.

Este patrón solicita al gestor de bases de datos que busque todos los valores que empiezan con WYSE y terminan con cinco espacios en blanco. Si tenía la intención de buscar sólo los valores que empiezan con 'WYSE', deberá asignar el valor 'WYSE% % % % %' a la variable del lenguaje principal.

Ejemplos

- Busque la serie 'SYSTEMS' que aparezca en la columna PROJNAME de la tabla PROJECT.

```
SELECT PROJNAME FROM PROJECT
WHERE PROJECT.PROJNAME LIKE '%SYSTEMS%'
```

- Busque una serie con un primer carácter 'J' que tenga exactamente dos caracteres de longitud en la columna FIRSTNAME de la tabla EMPLOYEE.

```
SELECT FIRSTNAME FROM EMPLOYEE
WHERE EMPLOYEE.FIRSTNAME LIKE 'J_'
```

- Busque una serie de caracteres, de cualquier longitud, cuyo primer carácter sea 'J', en la columna FIRSTNAME de la tabla EMPLOYEE.

Predicado LIKE

```
SELECT FIRSTNAME FROM EMPLOYEE
WHERE EMPLOYEE.FIRSTNAME LIKE 'J%'
```

- En la tabla CORP_SERVERS, busque una serie en la columna LA_SERVERS que coincida con el valor del registro especial CURRENT SERVER.

```
SELECT LA_SERVERS FROM CORP_SERVERS
WHERE CORP_SERVERS.LA_SERVERS LIKE CURRENT SERVER
```

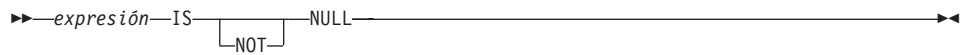
- Recupere todas las series que empiecen por la secuencia de caracteres '%_\' en la columna A de la tabla T.

```
SELECT A FROM T WHERE T.A LIKE
'%_\' ESCAPE '\'
```

- Utilice la función escapar BLOB para obtener un carácter de escape de un byte que sea compatible con los tipos de datos coincidentes y de patrón (ambos BLOB).

```
SELECT COLBLOB FROM TABLET
WHERE COLBLOB LIKE :patrón_var ESCAPE BLOB(X'0E')
```

Predicado NULL



El predicado NULL comprueba la existencia de valores nulos.

El resultado de un predicado NULL no puede ser desconocido. Si el valor de la expresión es nulo, el resultado es verdadero. Si el valor no es nulo, el resultado es falso. Si se especifica NOT, el resultado se invierte.

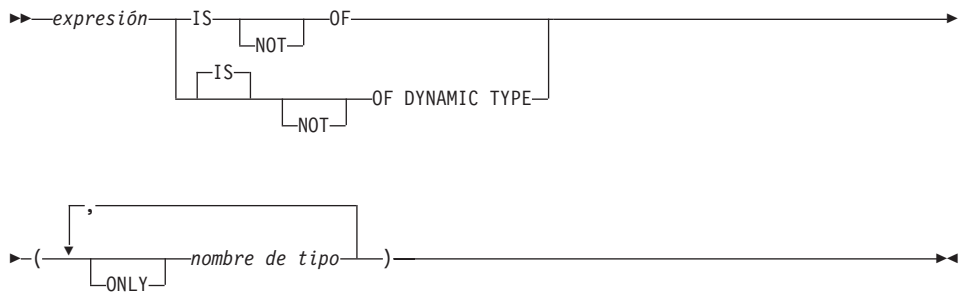
Ejemplos:

PHONENO **IS NULL**

SALARY **IS NOT NULL**

Predicado TYPE

Predicado TYPE



Un *predicado TYPE* compara el tipo de una expresión con uno o más tipos estructurados definidos por el usuario.

El tipo dinámico de una expresión que implica desreferenciar un tipo de referencia es el tipo real de la fila referenciada de la tabla o vista con tipo de destino. Puede diferenciarse del tipo de destino de una expresión que implica la referencia, denominado el tipo estático de la expresión.

Si el valor de *expresión* es nulo, el resultado del predicado será desconocido. El resultado del predicado será verdadero si el tipo dinámico de la *expresión* es un subtipo de uno de los tipos estructurados especificados por *nombre de tipo*; de lo contrario, el resultado será falso. Si **ONLY** precede cualquier *nombre de tipo*, no se tienen en cuenta los subtipos correspondientes de este tipo.

Si *nombre de tipo* no está calificado, se resuelve utilizando la vía de acceso de SQL. Cada *nombre de tipo* debe identificar un tipo definido por el usuario que esté en la jerarquía de tipos del tipo estático de *expresión* (SQLSTATE 428DU).

Debe utilizarse la función Deref siempre que el predicado TYPE tenga una expresión que implique un valor de tipo de referencia. El tipo estático de esta forma de *expresión* es el tipo de destino de la referencia. Consulte "Deref" en la página 317 para obtener más información sobre la función Deref.

La sintaxis IS OF y OF DYNAMIC TYPE son alternativas equivalentes para el predicado TYPE. Asimismo, IS NOT OF y NOT OF DYNAMIC TYPE son alternativas equivalentes.

Ejemplo:

Existe una jerarquía de tablas que tiene una tabla raíz EMPLOYEE de tipo EMP y una subtabla MANAGER de tipo MGR. Otra tabla, ACTIVITIES, incluye una columna denominada WHO_RESPONSIBLE que está definida

como REF(EMP) SCOPE EMPLOYEE. Lo siguiente es un predicado de tipo que devuelve un resultado verdadero cuando una fila correspondiente a WHO_RESPONSIBLE es un director ("manager"):

```
DEREF (WHO_RESPONSIBLE) IS OF (MGR)
```

Si una tabla contiene una columna EMPLOYEE de tipo EMP, EMPLOYEE puede contener valores de tipo EMP y también valores de sus subtipos, tales como MGR. El predicado siguiente

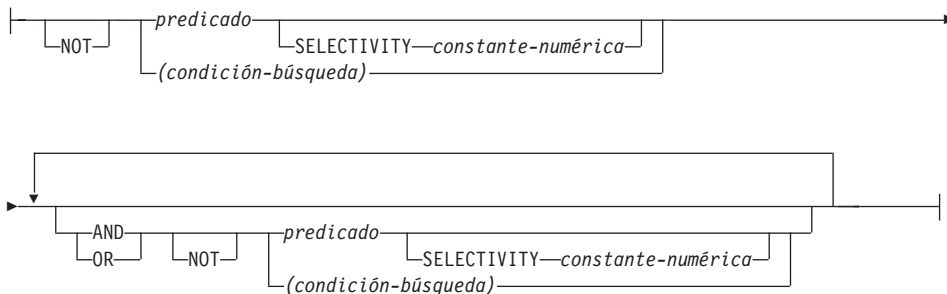
```
EMPL IS OF (MGR)
```

devuelve un resultado verdadero cuando EMPL no es nulo y es realmente un director.

Condiciones de búsqueda

Condiciones de búsqueda

condición-búsqueda:



Una *condición de búsqueda* especifica una condición que es “verdadera,” “falsa,” o “desconocida” acerca de una fila determinada.

El resultado de una condición de búsqueda se deriva por la aplicación de *operadores lógicos* (AND, OR, NOT) especificados al resultado de cada predicado especificado. Si no se especifican operadores lógicos, el resultado de la condición de búsqueda es el resultado del predicado especificado.

AND y OR se definen en la Tabla 14, en la que P y Q son unos predicados cualesquiera:

Tabla 14. Tablas de evaluación para AND y OR

P	Q	P AND Q	P OR Q
Verdadero	Verdadero	Verdadero	Verdadero
Verdadero	Falso	Falso	Verdadero
Verdadero	Desconocido	Desconocido	Verdadero
Falso	Verdadero	Falso	Verdadero
Falso	Falso	Falso	Falso
Falso	Desconocido	Falso	Desconocido
Desconocido	Verdadero	Desconocido	Verdadero
Desconocido	Falso	Falso	Desconocido
Desconocido	Desconocido	Desconocido	Desconocido

NOT(verdadero) es falso, NOT(falso) es verdadero y NOT(desconocido) es desconocido.

En primer lugar se evalúan las condiciones de búsqueda entre paréntesis. Si el orden de evaluación no se especifica mediante paréntesis, se aplica NOT antes de AND y AND se aplica antes de OR. El orden en el que se evalúan los operadores del mismo nivel de prioridad no está definido, para permitir la optimización de condiciones de búsqueda.

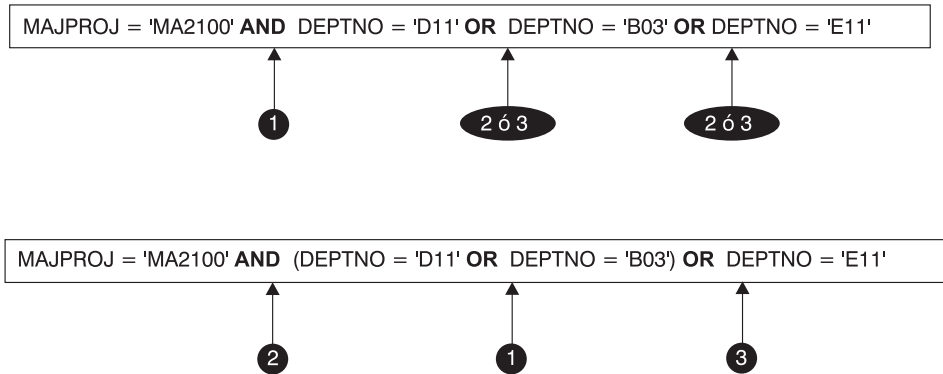


Figura 13. Orden de evaluación de las condiciones de búsqueda

SELECTIVITY *valor*

La cláusula SELECTIVITY se utiliza para indicar a DB2 qué porcentaje de selectividad prevista corresponde al predicado. SELECTIVITY se puede especificar sólo cuando el predicado es un predicado definido por el usuario.

Un predicado definido por el usuario consta de una invocación de función definida por el usuario, en el contexto de una especificación de predicado que coincide con la existente en la cláusula PREDICATES de CREATE FUNCTION. Por ejemplo, si la función foo está definida con PREDICATES WHEN=1..., es válido utilizar SELECTIVITY de este modo:

```
SELECT *
  FROM STORES
 WHERE foo(parm,parm) = 1 SELECTIVITY 0.004
```

El valor de selectividad debe ser un valor literal numérico comprendido dentro del rango inclusivo 0-1 (SQLSTATE 42615). Si SELECTIVITY no se especifica, el valor por omisión es 0.01 (es decir, el predicado definido por el usuario debe descartar todas las filas de la tabla excepto un 1 por ciento. El valor por omisión de SELECTIVITY se puede modificar para una función determinada actualizando su columna SELECTIVITY en la vista SYSSTAT.FUNCTIONS. Se obtiene un error si la cláusula SELECTIVITY se especifica para un predicado no definido por el usuario (SQLSTATE 428E5).

Condiciones de búsqueda

Se puede utilizar una función definida por el usuario (UDF) como predicado definido por el usuario y, por tanto, puede permitir la utilización de índices si:

- La especificación de predicado está presente en la sentencia CREATE FUNCTION
- la UDF se invoca en una cláusula WHERE que se compara (sintácticamente) de la misma manera que se especifica en la especificación de predicado
- no existe ninguna negación (operador NOT)

Ejemplos

En la consulta siguiente, la especificación UDF interna de la cláusula WHERE cumple las tres condiciones y se considera que es un predicado definido por el usuario. (Para obtener más información sobre las UDF de `within` y de `distance`, vea la sección de Ejemplos de “CREATE FUNCTION (Escalar externa)” en la página 696.)

```
SELECT *
FROM customers
WHERE within(location, :sanJose) = 1 SELECTIVITY 0.2
```

Sin embargo, la presencia de `within` en la consulta siguiente no permite el uso de índices debido a la negación, y no se considera un predicado definido por el usuario.

```
SELECT *
FROM customers
WHERE NOT(within(location, :sanJose) = 1) SELECTIVITY 0.3
```

En el ejemplo siguiente, se identifican los clientes y tiendas que están a una determinada distancia entre sí. La distancia de una tienda a otra se calcula mediante el radio de la ciudad donde viven los clientes.

```
SELECT *
FROM customers, stores
WHERE distance(customers.loc, stores.loc) <
CityRadius(stores.loc) SELECTIVITY 0.02
```

En la consulta anterior, se considera que el predicado contenido en la cláusula WHERE es un predicado definido por el usuario. El resultado producido por `CityRadius` se utiliza como argumento de búsqueda para la función productora de rangos.

Sin embargo, como el resultado devuelto por `CityRadius` se utiliza como función productora de rangos, el predicado definido por el usuario no podrá utilizar la extensión de índice definida para la columna `stores.loc`. Por lo tanto, la UDF sólo utilizará el índice definido en la columna `customers.loc`.

Capítulo 4. Funciones

Una función es una operación que se indica por un nombre de función seguido de un par de paréntesis que encierran la especificación de los argumentos (es posible que no haya ningún argumento).

Las funciones se clasifican en *funciones de columna*, *funciones escalares*, *funciones de fila* y *funciones de tabla*.

- El argumento de una función de columna es un conjunto de valores iguales. Devuelve un solo valor (posiblemente nulo) y se puede especificar en una sentencia de SQL donde se pueda utilizar una *expresión*. En la utilización de las funciones de columna se aplican restricciones adicionales que se especifican en el apartado "Funciones de columna" en la página 257.
- El argumento o argumentos de una función escalar son valores escalares individuales, que pueden tener diferentes tipos y distintos significados. Devuelve un solo valor (posiblemente nulo) y se puede especificar en una sentencia de SQL en la que pueda utilizarse una *expresión*.
- El argumento de una función de fila es un tipo estructurado. Esta función devuelve una fila de tipos de datos incorporados y sólo se puede especificar como función de transformación para un tipo estructurado.
- El argumento o argumentos de la función de tabla son valores escalares individuales, que pueden tener tipos diferentes y distintos significados. Devuelve una tabla a la sentencia de SQL, y sólo puede especificarse en la cláusula FROM de una sentencia SELECT. Se aplican restricciones adicionales a la utilización de funciones de tabla, tal como se especifica en el apartado "cláusula-from" en la página 478.

La Tabla 15 en la página 236 muestra las funciones a las que se da soporte. El "Nombre de función" combinado con el "Esquema" proporciona el nombre totalmente calificado de la función. "Descripción" describe brevemente lo que hace la función. "Parámetros de entrada" proporciona el tipo de datos esperado para cada argumento durante la invocación de función. Muchas funciones incluyen variaciones de los parámetros de entrada, que permiten utilizar diferentes tipos de datos o un número diferente de argumentos. La combinación del esquema, nombre de función y parámetros de entrada conforman una *signatura de función*. Cada *signatura de función* puede devolver un valor de un tipo distinto que se muestra en las columnas "Devuelve".

Existen algunas distinciones que deben entenderse acerca de los tipos de parámetros de entrada. En algunos casos el tipo se especifica como un tipo de datos incorporado específico y en otros casos utilizará una variable general

Funciones

como *cualquier-tipo-numérico*. Cuando se lista un tipo de datos específico, significa que una coincidencia exacta sólo aparecerá con el tipo de datos especificado. Cuando se utiliza una variable general, cada uno de los tipos de datos asociados con dicha variable dará como resultado una coincidencia exacta. Esta distinción afecta a la selección de función como se describe en el apartado “Resolución de funciones” en la página 162.

Puede haber funciones adicionales disponibles porque las funciones definidas por el usuario pueden crearse en esquemas diferentes utilizando una de estas firmas de función como fuente (consulte el apartado “CREATE FUNCTION” en la página 695 para obtener detalles) o los usuarios pueden crear funciones externas utilizando sus propios programas.

Nota:

- Junto con el gestor de bases de datos se proporcionan funciones incorporadas, que devuelven un valor resultante individual y están catalogadas como parte del esquema SYSIBM. Ejemplos de dichas funciones son las funciones de columna, como AVG, funciones de operador, como “+”, funciones de conversión, como DECIMAL y otras como SUBSTR.
- Las funciones definidas por el usuario son funciones que están registradas en una base de datos en SYSCAT.FUNCTIONS (utilizando la sentencia CREATE FUNCTION). Estas funciones nunca forman parte del esquema SYSIBM. El gestor de bases de datos proporciona un conjunto de estas funciones dentro de un esquema denominado SYSFUN.

Tabla 15. Funciones soportadas

Nombre de función	Esquema	Descripción	
	Parámetros de entrada		Devuelve
ABS o ABSVAL	SYSIBM	Devuelve el valor absoluto del argumento.	
	Cualquier expresión que devuelve un tipo de datos numérico incorporado.		Mismo tipo de datos y longitud que el argumento
ABS o ABSVAL	SYSFUN	Devuelve el valor absoluto del argumento.	
	SMALLINT		SMALLINT
	INTEGER		INTEGER
	BIGINT		BIGINT
	DOUBLE		DOUBLE
ACOS	SYSFUN	Devuelve el arco coseno del argumento, en forma de ángulo expresado en radianes.	
	DOUBLE		DOUBLE

Tabla 15. Funciones soportadas (continuación)

Nombre de función	Esquema	Descripción
	Parámetros de entrada	
ASCII	SYSFUN	Devuelve el valor de código ASCII del carácter más a la izquierda del argumento, expresado en forma de entero.
	CHAR	INTEGER
	VARCHAR(4000)	INTEGER
	CLOB(1M)	INTEGER
ASIN	SYSFUN	Devuelve el arco seno del argumento, en forma de ángulo expresado en radianes.
	DOUBLE	DOUBLE
ATAN	SYSFUN	Devuelve el arco tangente del argumento, en forma de ángulo expresado en radianes.
	DOUBLE	DOUBLE
ATAN2	SYSFUN	Devuelve el arco tangente de las coordenadas x e y , especificadas por el primer y segundo argumentos respectivamente, en forma de ángulo expresado en radianes.
	DOUBLE, DOUBLE	DOUBLE
AVG	SYSIBM	Devuelve el promedio de un conjunto de números (función de columna).
	<i>tipo-numérico</i> ⁴	<i>tipo-numérico</i> ¹
BIGINT	SYSIBM	Devuelve una representación de entero de 64 bits de un número o serie de caracteres en el formato de una constante de enteros.
	<i>tipo-numérico</i>	BIGINT
	VARCHAR	BIGINT
BLOB	SYSIBM	Convierte el tipo fuente a BLOB, con una longitud opcional.
	<i>tipo-serie</i>	BLOB
	<i>tipo-serie</i> , INTEGER	BLOB
CEIL o CEILING	SYSFUN	Devuelve el entero más pequeño que es mayor o igual al argumento.
	SMALLINT	SMALLINT
	INTEGER	INTEGER
	BIGINT	BIGINT
	DOUBLE	DOUBLE

Funciones

Tabla 15. Funciones soportadas (continuación)

Nombre de función	Esquema	Descripción	
	Parámetros de entrada		Devuelve
CHAR	SYSIBM	Devuelve una representación de serie del tipo fuente.	
	<i>tipo-caracteres</i>		CHAR
	<i>tipo-caracteres</i> , INTEGER		CHAR(entero)
	<i>tipo-fechahora</i>		CHAR
	<i>tipo-fechahora</i> , <i>palabraclave</i> ²		CHAR
	SMALLINT		CHAR(6)
	INTEGER		CHAR(11)
	BIGINT		CHAR(20)
	DECIMAL		CHAR(2+precisión)
	DECIMAL, VARCHAR		CHAR(2+precisión)
CHAR	SYSFUN	Devuelve una representación de serie de caracteres de un número de coma flotante.	
	DOUBLE		CHAR(24)
CHR	SYSFUN	Devuelve el carácter que tiene el valor de código ASCII especificado por el argumento. El valor del argumento debe estar entre 0 y 255; de lo contrario, el valor de retorno es nulo.	
	INTEGER		CHAR(1)
CLOB	SYSIBM	Convierte el tipo fuente a CLOB, con una longitud opcional.	
	<i>tipo-caracteres</i>		CLOB
	<i>tipo-caracteres</i> , INTEGER		CLOB
COALESCE ³	SYSIBM	Devuelve el primer argumento que no es nulo del conjunto de argumentos.	
	<i>cualquier-tipo</i> , <i>cualquier-tipo-compatible-unión</i> , ...		<i>cualquier-tipo</i>
CONCAT o	SYSIBM	Devuelve la concatenación de 2 argumentos de serie.	
	<i>tipo-serie</i> , <i>tipo-serie-compatible</i>		<i>tipo-serie máx</i>
CORRELATION o CORR	SYSIBM	Devuelve el coeficiente de correlación de un conjunto de pares de números.	
	<i>tipo-numérico</i> , <i>tipo-numérico</i>		DOUBLE
COS	SYSFUN	Devuelve el coseno del argumento, donde el argumento es un ángulo expresado en radianes.	
	DOUBLE		DOUBLE
COT	SYSFUN	Devuelve la cotangente del argumento, donde el argumento es un ángulo expresado en radianes.	
	DOUBLE		DOUBLE
COUNT	SYSIBM	Devuelve la cuenta del número de filas en un conjunto de filas o valores (función de columna).	
	<i>cualquier-tipo-incorporado</i> ⁴		INTEGER

Tabla 15. Funciones soportadas (continuación)

Nombre de función	Esquema	Descripción	
	Parámetros de entrada		Devuelve
COUNT_BIG	SYSIBM	Devuelve el número de filas o de valores de un conjunto de filas o de valores (función de columna). El resultado puede ser mayor que el valor máximo de entero.	
	<i>cualquier-tipo-incorporado</i> ⁴		DECIMAL(31,0)
COVARIANCE o COVAR	SYSIBM	Devuelve la covarianza de un conjunto de pares de números.	
	<i>tipo-numérico, tipo-numérico</i>		DOUBLE
DATE	SYSIBM	Devuelve una fecha de un solo valor de entrada.	
	DATE		DATE
	TIMESTAMP		DATE
	DOUBLE		DATE
	VARCHAR		DATE
DAY	SYSIBM	Devuelve la parte correspondiente al día de un valor.	
	VARCHAR		INTEGER
	DATE		INTEGER
	TIMESTAMP		INTEGER
	DECIMAL		INTEGER
DAYNAME	SYSFUN	Devuelve una serie de caracteres en mayúsculas y minúsculas mezcladas que contiene el nombre del día (por ejemplo viernes) correspondiente a la parte de día del argumento basándose en el entorno nacional existente al emitirse db2start.	
	VARCHAR(26)		VARCHAR(100)
	DATE		VARCHAR(100)
	TIMESTAMP		VARCHAR(100)
DAYOFWEEK	SYSFUN	Devuelve el día de la semana del argumento, en forma de valor entero comprendido dentro del rango 1-7, donde 1 representa el domingo.	
	VARCHAR(26)		INTEGER
	DATE		INTEGER
	TIMESTAMP		INTEGER
DAYOFWEEK_ISO	SYSFUN	Devuelve el día de la semana del argumento, en forma de valor entero comprendido dentro del rango 1-7, donde 1 representa el lunes.	
	VARCHAR(26)		INTEGER
	DATE		INTEGER
	TIMESTAMP		INTEGER
DAYOFYEAR	SYSFUN	Devuelve el día del año del argumento como un valor entero en el rango de 1 a 366.	
	VARCHAR(26)		INTEGER
	DATE		INTEGER
	TIMESTAMP		INTEGER

Funciones

Tabla 15. Funciones soportadas (continuación)

Nombre de función	Esquema	Descripción	
	Parámetros de entrada		Devuelve
DAYS	SYSIBM	Devuelve una representación de entero de una fecha.	
	VARCHAR		INTEGER
	TIMESTAMP		INTEGER
	DATE		INTEGER
DBCLOB	SYSIBM	Convierte el tipo fuente a DBCLOB, con una longitud opcional.	
	<i>tipo-gráfico</i>		DBCLOB
	<i>tipo-gráfico</i> , INTEGER		DBCLOB
DECIMAL o DEC	SYSIBM	Devuelve la representación decimal de un número, con la precisión y escala opcionales.	
	<i>tipo-numérico</i>		DECIMAL
	<i>tipo-numérico</i> , INTEGER		DECIMAL
	<i>tipo-numérico</i> INTEGER, INTEGER		DECIMAL
DECIMAL o DEC	SYSIBM	Devuelve la representación decimal de una serie de caracteres, con precisión, escala y caracteres decimales opcionales.	
	VARCHAR		DECIMAL
	VARCHAR, INTEGER		DECIMAL
	VARCHAR, INTEGER, INTEGER		DECIMAL
	VARCHAR, INTEGER, INTEGER, VARCHAR		DECIMAL
DECRYPT_BIN	SYSIBM	Devuelve un valor que es el resultado del descifrado de datos cifrados utilizando una serie de contraseña.	
	VARCHAR FOR BIT DATA		VARCHAR FOR BIT DATA
	VARCHAR FOR BIT DATA, VARCHAR		VARCHAR FOR BIT DATA
DECRYPT_CHAR	SYSIBM	Devuelve un valor que es el resultado del descifrado de datos cifrados utilizando una serie de contraseña.	
	VARCHAR FOR BIT DATA		VARCHAR
	VARCHAR FOR BIT DATA, VARCHAR		VARCHAR
DEGREES	SYSFUN	Devuelve el número de grados convertidos del argumento expresados en radianes.	
	DOUBLE		DOUBLE
DEREF	SYSIBM	Devuelve una instancia del tipo de destino del argumento del tipo de referencia.	
	REF(<i>cualquier-tipo-estructurado</i>) con ámbito definido		<i>cualquier-tipo-estructurado</i> (igual que el tipo de destino de entrada)
DIFFERENCE	SYSFUN	Devuelve la diferencia entre los sonidos de las palabras de las dos series argumento, tal como esa diferencia se determina mediante la utilización de la función SOUNDEX. Un valor de 4 significa que las series tienen igual sonido.	
	VARCHAR(4000), VARCHAR(4000)		INTEGER

Tabla 15. Funciones soportadas (continuación)

Nombre de función	Esquema	Descripción	
	Parámetros de entrada		Devuelve
DIGITS	SYSIBM	Devuelve la representación de serie de caracteres de un número.	
	DECIMAL		CHAR
DLCOMMENT	SYSIBM	Devuelve el atributo del comentario de un valor DATALINK.	
	DATALINK		VARCHAR(254)
DLLINKTYPE	SYSIBM	Devuelve el atributo del tipo de enlace de un valor DATALINK.	
	DATALINK		VARCHAR(4)
DLURLCOMPLETE	SYSIBM	Devuelve el URL completo (incluido el símbolo de acceso) de un valor DATALINK.	
	DATALINK		VARCHAR
DLURLPATH	SYSIBM	Devuelve la vía de acceso y el nombre de archivo (incluido el símbolo de acceso) de un valor DATALINK.	
	DATALINK		VARCHAR
DLURLPATHONLY	SYSIBM	Devuelve la vía de acceso y el nombre de archivo (sin símbolo de accesos) de un valor DATALINK.	
	DATALINK		VARCHAR
DLURLSCHEME	SYSIBM	Devuelve el esquema del atributo del URL de un valor DATALINK.	
	DATALINK		VARCHAR
DLURLSERVER	SYSIBM	Devuelve el servidor del atributo del URL de un valor DATALINK.	
	DATALINK		VARCHAR
DLVALUE	SYSIBM	Crea un valor DATALINK a partir de un argumento ubicación-datos, un argumento tipo de enlace y un argumento serie-comentario opcional.	
	VARCHAR		DATALINK
	VARCHAR, VARCHAR		DATALINK
	VARCHAR, VARCHAR, VARCHAR		DATALINK
DOUBLE o DOUBLE_PRECISION	SYSIBM	Devuelve la representación en coma flotante de un número.	
	<i>tipo-numérico</i>		DOUBLE
DOUBLE	SYSFUN	Devuelve el número de coma flotante, correspondiente a la representación de serie de caracteres de un número. Se ignoran los blancos iniciales y de cola de <i>argumento</i> .	
	VARCHAR		DOUBLE
ENCRYPT	SYSIBM	Devuelve un valor que es el resultado del descifrado de una expresión de serie de datos.	
	VARCHAR		VARCHAR FOR BIT DATA
	VARCHAR, VARCHAR		VARCHAR FOR BIT DATA
	VARCHAR, VARCHAR, VARCHAR		VARCHAR FOR BIT DATA

Funciones

Tabla 15. Funciones soportadas (continuación)

Nombre de función	Esquema	Descripción	
	Parámetros de entrada		Devuelve
EVENT_MON_STATE	SYSIBM	Devuelve el estado operativo de un supervisor de sucesos en particular.	
	VARCHAR		INTEGER
EXP	SYSFUN	Devuelve la función exponencial del argumento.	
	DOUBLE		DOUBLE
FLOAT	SYSIBM	Igual que DOUBLE.	
FLOOR	SYSFUN	Devuelve el valor de entero más grande que es menor o igual al argumento.	
	SMALLINT		SMALLINT
	INTEGER		INTEGER
	BIGINT		BIGINT
	DOUBLE		DOUBLE
GETHINT	SYSIBM	Devuelve la indicación de contraseña si se encuentra alguna.	
	VARCHAR o CLOB		VARCHAR
GENERATE_UNIQUE	SYSIBM	Devuelve una serie de caracteres de datos de bits que es exclusiva comparada a cualquier otra ejecución de la misma función.	
	<i>ningún argumento</i>		CHAR(13) FOR BIT DATA
GET_ROUTINE_SAR	SYSFUN	Devuelve la información necesaria para instalar una rutina idéntica en otro servidor de bases de datos que se ejecute al mismo nivel y ejecute el mismo sistema operativo.	
	BLOB(3M), CHAR(2), VARCHAR(257)		BLOB(3M)
GRAPHIC	SYSIBM	Convierte el tipo fuente a GRAPHIC, con una longitud opcional.	
	<i>tipo-gráfico</i>		GRAPHIC
	<i>tipo-gráfico</i> , INTEGER		GRAPHIC
GROUPING	SYSIBM	Se utiliza con conjuntos-agrupaciones y supergrupos para indicar filas de subtotales generadas por un conjunto de agrupaciones (función de columna). El valor devuelto es:	
		<p>1 El valor del argumento de la fila devuelta es un valor nulo y la fila se ha generado para un conjunto de agrupaciones. Esta fila generada proporciona un subtotal para un conjunto de agrupaciones.</p> <p>0 en otro caso.</p>	
	<i>cualquier-tipo</i>		SMALLINT
HEX	SYSIBM	Devuelve la representación hexadecimal de un valor.	
	<i>cualquier-tipo-incorporado</i>		VARCHAR

Tabla 15. Funciones soportadas (continuación)

Nombre de función	Esquema	Descripción	
	Parámetros de entrada		Devuelve
HOUR	SYSIBM	Devuelve la parte correspondiente a la hora de un valor.	
	VARCHAR		INTEGER
	TIME		INTEGER
	TIMESTAMP		INTEGER
	DECIMAL		INTEGER
IDENTITY_VAL_LOCAL	SYSIBM	Devuelve el valor asignado más recientemente para una columna de identidad.	
			DECIMAL
INSERT	SYSFUN	Devuelve una serie donde se han suprimido <i>argumento3</i> bytes de <i>argumento1</i> empezando en <i>argumento2</i> y donde <i>argumento4</i> se ha insertado en <i>argumento1</i> empezando en <i>argumento2</i> .	
	VARCHAR(4000), INTEGER, INTEGER, VARCHAR(4000)		VARCHAR(4000)
	CLOB(1M), INTEGER, INTEGER, CLOB(1M)		CLOB(1M)
	BLOB(1M), INTEGER, INTEGER, BLOB(1M)		BLOB(1M)
INTEGER o INT	SYSIBM	Devuelve la representación de entero de un número.	
	<i>tipo-numérico</i>		INTEGER
	VARCHAR		INTEGER
JULIAN_DAY	SYSFUN	Devuelve un valor de entero que representa el número de días desde el 1 de enero de 4712 A.C. (el inicio del calendario Juliano) hasta el valor de fecha especificado en el <i>argumento</i> .	
	VARCHAR(26)		INTEGER
	DATE		INTEGER
	TIMESTAMP		INTEGER
LCASE o LOWER	SYSIBM	Devuelve una serie en la que todos los caracteres se han convertido a minúsculas.	
	CHAR		CHAR
	VARCHAR		VARCHAR
LCASE	SYSFUN	Devuelve una serie en la que todos los caracteres se han convertido a minúsculas. LCASE sólo manejará caracteres del conjunto no variante. Por lo tanto, LCASE(UCASE(serie)) no devolverá necesariamente el mismo resultado que LCASE(serie).	
	VARCHAR(4000)		VARCHAR(4000)
	CLOB(1M)		CLOB(1M)
LEFT	SYSFUN	Devuelve una serie que consta de los <i>argumento2</i> bytes más a la izquierda del <i>argumento1</i> .	
	VARCHAR(4000), INTEGER		VARCHAR(4000)
	CLOB(1M), INTEGER		CLOB(1M)
	BLOB(1M), INTEGER		BLOB(1M)

Funciones

Tabla 15. Funciones soportadas (continuación)

Nombre de función	Esquema	Descripción	Devuelve
	Parámetros de entrada		
LENGTH	SYSIBM	Devuelve la longitud del operando en bytes (excepto los tipos de serie de doble byte que devuelven la longitud en caracteres).	
	<i>cualquier-tipo-incorporado</i>		INTEGER
LN	SYSFUN	Devuelve el logaritmo natural del argumento (igual que LOG).	
	DOUBLE		DOUBLE
LOCATE	SYSFUN	Devuelve la posición inicial de la primera ocurrencia del <i>argumento1</i> en el <i>argumento2</i> . Si se especifica el tercer argumento opcional, indica la posición del carácter en el <i>argumento2</i> en el que la búsqueda tiene que empezar. Si el <i>argumento1</i> no se encuentra en el <i>argumento2</i> , se devuelve el valor 0.	
	VARCHAR(4000), VARCHAR(4000)		INTEGER
	VARCHAR(4000), VARCHAR(4000), INTEGER		INTEGER
	CLOB(1M), CLOB(1M)		INTEGER
	CLOB(1M), CLOB(1M), INTEGER		INTEGER
	BLOB(1M), BLOB(1M)		INTEGER
LOG	SYSFUN	Devuelve el logaritmo natural del argumento (igual que LN).	
	DOUBLE		DOUBLE
LOG10		Devuelve el logaritmo de base 10 del argumento.	
	DOUBLE		DOUBLE
LONG_VARCHAR	SYSIBM	Devuelve una serie larga.	
	<i>tipo-caracteres</i>		LONG VARCHAR
LONG_VARGRAPHIC	SYSIBM	Convierte el tipo fuente a LONG_VARGRAPHIC.	
	<i>tipo-gráfico</i>		LONG VARGRAPHIC
LTRIM	SYSIBM	Devuelve los caracteres del argumento habiendo eliminado los blancos iniciales.	
	CHAR		VARCHAR
	VARCHAR		VARCHAR
	GRAPHIC		VARGRAPHIC
LTRIM	VARGRAPHIC		VARGRAPHIC
	SYSFUN	Devuelve los caracteres del argumento habiendo eliminado los blancos iniciales.	
	VARCHAR(4000)		VARCHAR(4000)
	CLOB(1M)		CLOB(1M)
MAX	SYSIBM	Devuelve el valor máximo de un conjunto de valores (función de columna).	
	<i>cualquier-tipo-incorporado</i> ⁵		<i>igual que un tipo de entrada</i>

Tabla 15. Funciones soportadas (continuación)

Nombre de función	Esquema	Descripción	
	Parámetros de entrada		Devuelve
MICROSECOND	SYSIBM	Devuelve la parte correspondiente a los microsegundos (unidad-tiempo) de un valor.	
	VARCHAR		INTEGER
	TIMESTAMP		INTEGER
	DECIMAL		INTEGER
MIDNIGHT_SECONDS	SYSFUN	Devuelve un valor entero en el rango de 0 a 86.400 que representa el número de segundos entre la medianoche y el valor de hora especificado en el <i>argumento</i> .	
	VARCHAR(26)		INTEGER
	TIME		INTEGER
	TIMESTAMP		INTEGER
MIN	SYSIBM	Devuelve el valor mínimo de un conjunto de valores (función de columna).	
	<i>cualquier-tipo-incorporado</i> ⁵		<i>igual que un tipo de entrada</i>
MINUTE	SYSIBM	Devuelve la parte correspondiente a los minutos de un valor.	
	VARCHAR		INTEGER
	TIME		INTEGER
	TIMESTAMP		INTEGER
	DECIMAL		INTEGER
MOD	SYSFUN	Devuelve el resto (módulos) del <i>argumento1</i> dividido por el <i>argumento2</i> . El resultado sólo es negativo si el <i>argumento1</i> es negativo.	
	SMALLINT, SMALLINT		SMALLINT
	INTEGER, INTEGER		INTEGER
	BIGINT, BIGINT		BIGINT
MONTH	SYSIBM	Devuelve la parte correspondiente al mes de un valor.	
	VARCHAR		INTEGER
	DATE		INTEGER
	TIMESTAMP		INTEGER
	DECIMAL		INTEGER
MONTHNAME	SYSFUN	Devuelve una serie de caracteres en mayúsculas y minúsculas mezcladas que contiene el nombre del mes (por ejemplo enero) correspondiente a la parte de mes del argumento que es una fecha o una indicación de la fecha y hora, basándose en el entorno nacional existente al iniciar la base de datos.	
	VARCHAR(26)		VARCHAR(100)
	DATE		VARCHAR(100)
	TIMESTAMP		VARCHAR(100)

Funciones

Tabla 15. Funciones soportadas (continuación)

Nombre de función	Esquema	Descripción	
	Parámetros de entrada		Devuelve
MQPUBLISH	MQDB2	Publica datos en una ubicación MQSeries.	
	VARCHAR(4000)		INTEGER
MQREAD	MQDB2	Devuelve un mensaje de una ubicación MQSeries.	
	<i>tipo-serie</i>		VARCHAR(4000)
MQREADALL	MQDB2	Devuelve una tabla con mensajes y metadatos de mensaje de una ubicación MQSeries.	
	Vea “MQREADALL” en la página 458.		
MQRECEIVE	MQDB2	Devuelve un mensaje de una ubicación MQSeries y elimina el mensaje de la cola asociada.	
	<i>tipo-serie</i>		VARCHAR(4000)
MQRECEIVEALL	MQDB2	Devuelve una tabla que contiene los mensajes y metadatos de mensaje de una ubicación MQSeries y elimina los mensajes de la cola asociada.	
	Vea “MQRECEIVEALL” en la página 461		
MQSEND	MQDB2	Envía datos a una ubicación MQSeries.	
	VARCHAR(4000)		INTEGER
MQSUBSCRIBE	MQDB2	Abona a los mensajes de MQSeries publicados acerca de un tema específico.	
	<i>tipo-serie</i>		INTEGER
MQUNSUBSCRIBE	MQDB2	Elimina el abono a los mensajes de MQSeries publicados acerca de un tema específico.	
	<i>tipo-serie</i>		INTEGER
MULTIPLY_ALT	SYSIBM	Devuelve el producto de dos argumentos como un valor decimal. Esta función es útil cuando la suma de las precisiones de argumento es superior a 31.	
	<i>tipo-numérico-exacto, tipo-numérico-exacto</i>		DECIMAL
NODENUMBER ³	SYSIBM	Devuelve el número de nodo de la fila. El argumento es un nombre de columna dentro de una tabla.	
	<i>cualquier-tipo</i>		INTEGER
NULLIF ³	SYSIBM	Devuelve NULL si los argumentos son igual, de lo contrario devuelve el primer argumento.	
	<i>cualquier-tipo</i> ⁵ , <i>cualquier-tipo-comparable</i> ⁵		<i>cualquier-tipo</i>
PARTITION ³	SYSIBM	Devuelve el índice de mapa de particionamiento (de 0 a 4095) de la fila. El argumento es un nombre de columna dentro de una tabla.	
	<i>cualquier-tipo</i>		INTEGER
POSSTR	SYSIBM	Devuelve la posición en la que una serie está contenida en otra.	
	<i>tipo-serie, tipo-serie-compatible</i>		INTEGER

Tabla 15. Funciones soportadas (continuación)

Nombre de función	Esquema	Descripción	
	Parámetros de entrada		Devuelve
POWER	SYSFUN	Devuelve el valor del <i>argumento1</i> elevado a la potencia del <i>argumento2</i> .	
	INTEGER, INTEGER		INTEGER
	BIGINT, BIGINT		BIGINT
	DOUBLE, INTEGER		DOUBLE
	DOUBLE, DOUBLE		DOUBLE
PUT_ROUTINE_SAR	SYSFUN	Pasa la información necesaria para crear y definir una rutina SQL en el servidor de bases de datos.	
	BLOB(3M)		
	BLOB(3M), VARCHAR(128), INTEGER		
QUARTER	SYSFUN	Devuelve un valor entero en el rango de 1 a 4 que representa el trimestre del año para la fecha especificada en el argumento.	
	VARCHAR(26)		INTEGER
	DATE		INTEGER
	TIMESTAMP		INTEGER
RADIANS	SYSFUN	Devuelve el número de radianes convertidos del argumento que se expresa en grados.	
	DOUBLE		DOUBLE
RAISE_ERROR ³	SYSIBM	Genera un error en la SQLCA. El sqlstate devuelto se indica por el <i>argumento1</i> . El segundo argumento contiene cualquier texto que se ha de devolver.	
	VARCHAR, VARCHAR		<i>cualquier-tipo</i> ⁶
RAND	SYSFUN	Devuelve un valor aleatorio de coma flotante comprendido entre 0 y 1, utilizando el argumento como valor generador opcional.	
	<i>ningún argumento necesario</i>		DOUBLE
	INTEGER		DOUBLE
REAL	SYSIBM	Devuelve la representación de coma flotante de precisión simple correspondiente a un número.	
	<i>tipo-numérico</i>		REAL
REC2XML	SYSIBM	Devuelve una serie formateada con códigos XML y que contiene nombres de columna y datos de columna.	
	DECIMAL, VARCHAR, VARCHAR, <i>cualquier-tipo</i> ⁷		VARCHAR
REGR_AVGX	SYSIBM	Devuelve cantidades que se utilizan para calcular estadísticas de diagnósticos.	
	<i>tipo-numérico, tipo-numérico</i>		DOUBLE
REGR_AVGY	SYSIBM	Devuelve cantidades que se utilizan para calcular estadísticas de diagnósticos.	
	<i>tipo-numérico, tipo-numérico</i>		DOUBLE

Funciones

Tabla 15. Funciones soportadas (continuación)

Nombre de función	Esquema	Descripción	
	Parámetros de entrada		Devuelve
REGR_COUNT	SYSIBM	Devuelve el número de pares de números no nulos que se utilizan para acomodar la línea de regresión.	
	<i>tipo-numérico, tipo-numérico</i>		INTEGER
REGR_INTERCEPT o REGR_ICPT	SYSIBM	Devuelve la intersección y de la línea de regresión.	
	<i>tipo-numérico, tipo-numérico</i>		DOUBLE
REGR_R2	SYSIBM	Devuelve el coeficiente de determinación de la regresión.	
	<i>tipo-numérico, tipo-numérico</i>		DOUBE
REGR_SLOPE	SYSIBM	Devuelve la inclinación de la línea.	
	<i>tipo-numérico, tipo-numérico</i>		DOUBLE
REGR_SXX	SYSIBM	Devuelve cantidades que se utilizan para calcular estadísticas de diagnósticos.	
	<i>tipo-numérico, tipo-numérico</i>		DOUBLE
REGR_SXY	SYSIBM	Devuelve cantidades que se utilizan para calcular estadísticas de diagnósticos.	
	<i>tipo-numérico, tipo-numérico</i>		DOUBLE
REGR_SYY	SYSIBM	Devuelve cantidades que se utilizan para calcular estadísticas de diagnósticos.	
	<i>tipo-numérico, tipo-numérico</i>		DOUBLE
REPEAT	SYSFUN	Devuelve una serie de caracteres compuesta del <i>argumento1</i> repetido <i>argumento2</i> veces.	
	VARCHAR(4000), INTEGER		VARCHAR(4000)
	CLOB(1M), INTEGER		CLOB(1M)
	BLOB(1M), INTEGER		BLOB(1M)
REPLACE	SYSFUN	Sustituye todas las ocurrencias del <i>argumento2</i> en el <i>argumento1</i> por el <i>argumento3</i> .	
	VARCHAR(4000), VARCHAR(4000), VARCHAR(4000)		VARCHAR(4000)
	CLOB(1M), CLOB(1M), CLOB(1M)		CLOB(1M)
	BLOB(1M), BLOB(1M), BLOB(1M)		BLOB(1M)
RIGHT	SYSFUN	Devuelve una serie que consta de los <i>argumento2</i> bytes más a la derecha del <i>argumento1</i> .	
	VARCHAR(4000), INTEGER		VARCHAR(4000)
	CLOB(1M), INTEGER		CLOB(1M)
	BLOB(1M), INTEGER		BLOB(1M)

Tabla 15. Funciones soportadas (continuación)

Nombre de función	Esquema	Descripción	
	Parámetros de entrada		Devuelve
ROUND	SYSFUN	Devuelve el primer argumento redondeado a <i>argumento2</i> posiciones a la derecha de la coma decimal. Si <i>argumento2</i> es negativo, <i>argumento1</i> se redondea al valor absoluto de <i>argumento2</i> posiciones a la izquierda de la coma decimal.	
	INTEGER, INTEGER		INTEGER
	BIGINT, INTEGER		BIGINT
	DOUBLE, INTEGER		DOUBLE
RTRIM	SYSIBM	Devuelve los caracteres del argumento habiendo eliminado los blancos de cola.	
	CHAR		VARCHAR
	VARCHAR		VARCHAR
	GRAPHIC		VARGRAPHIC
	VARGRAPHIC		VARGRAPHIC
RTRIM	SYSFUN	Devuelve los caracteres del argumento habiendo eliminado los blancos de cola.	
	VARCHAR(4000)		VARCHAR(4000)
	CLOB(1M)		CLOB(1M)
SECOND	SYSIBM	Devuelve la parte correspondiente a los segundos (unidad-tiempo) de un valor.	
	VARCHAR		INTEGER
	TIME		INTEGER
	TIMESTAMP		INTEGER
	DECIMAL		INTEGER
SIGN	SYSFUN	Devuelve un indicador del signo del argumento. Si el argumento es menor que cero, se devuelve -1. Si el argumento es igual a cero, devuelve 0. Si el argumento es mayor que cero, devuelve 1.	
	SMALLINT		SMALLINT
	INTEGER		INTEGER
	BIGINT		BIGINT
	DOUBLE		DOUBLE
SIN	SYSFUN	Devuelve el seno del argumento, donde el argumento es un ángulo expresado en radianes.	
	DOUBLE		DOUBLE
SMALLINT	SYSIBM	Devuelve una representación de entero pequeño de un número.	
	<i>tipo-numérico</i>		SMALLINT
	VARCHAR		SMALLINT

Funciones

Tabla 15. Funciones soportadas (continuación)

Nombre de función	Esquema	Descripción	
	Parámetros de entrada		Devuelve
SOUNDEX	SYSFUN	Devuelve un código de 4 caracteres que representa el sonido de las palabras del argumento. El resultado se puede utilizar para compararlo con el sonido de otras series. Consulte también DIFFERENCE.	
	VARCHAR(4000)		CHAR(4)
SPACE	SYSFUN	Devuelve una serie de caracteres que consta de <i>argumento1</i> espacios en blanco.	
	INTEGER		VARCHAR(4000)
SQLCACHE_SNAPSHOT	SYSFUN	Devuelve una tabla de la instantánea de la antememoria de sentencias de SQL dinámicas de db2 (función de tabla).	
	Consulte "SQLCACHE_SNAPSHOT" en la página 464.		
SQRT	SYSFUN	Devuelve la raíz cuadrada del argumento.	
	DOUBLE		DOUBLE
STDDEV	SYSIBM	Devuelve la desviación estándar de un conjunto de números (función de columna).	
	DOUBLE		DOUBLE
SUBSTR	SYSIBM	Devuelve una subserie de una serie <i>argumento1</i> empezando en el <i>argumento2</i> de <i>argumento3</i> caracteres. Si no se especifica el <i>argumento3</i> , se supone el resto de la serie.	
	<i>tipo-serie</i> , INTEGER		<i>tipo-serie</i>
	<i>tipo-serie</i> , INTEGER, INTEGER		<i>tipo-serie</i>
SUM	SYSIBM	Devuelve la suma de un conjunto de números (función de columna).	
	<i>tipo-numérico</i> ⁴		<i>tipo-numérico-máx</i> ¹
TABLE_NAME	SYSIBM	Devuelve un nombre no calificado de una tabla o vista basado en el nombre de objeto dado en el <i>argumento1</i> y el nombre de esquema opcional dado en el <i>argumento2</i> . Se utiliza para resolver los seudónimos.	
	VARCHAR		VARCHAR(128)
	VARCHAR, VARCHAR		VARCHAR(128)
TABLE_SCHEMA	SYSIBM	Devuelve la parte correspondiente al nombre de esquema del nombre de tabla o vista de dos partes dado por el nombre del objeto del <i>argumento1</i> y el nombre de esquema opcional del <i>argumento2</i> . Se utiliza para resolver los seudónimos.	
	VARCHAR		VARCHAR(128)
	VARCHAR, VARCHAR		VARCHAR(128)
TAN	SYSFUN	Devuelve la tangente del argumento, donde el argumento es un ángulo expresado en radianes.	
	DOUBLE		DOUBLE

Tabla 15. Funciones soportadas (continuación)

Nombre de función	Esquema	Descripción	
	Parámetros de entrada		Devuelve
TIME	SYSIBM	Devuelve una hora de un valor.	
	TIME		TIME
	TIMESTAMP		TIME
	VARCHAR		TIME
TIMESTAMP	SYSIBM	Devuelve una indicación de la hora de un valor o de un par de valores.	
	TIMESTAMP		TIMESTAMP
	VARCHAR		TIMESTAMP
	VARCHAR, VARCHAR		TIMESTAMP
	VARCHAR, TIME		TIMESTAMP
	DATE, VARCHAR		TIMESTAMP
DATE, TIME		TIMESTAMP	
TIMESTAMP_ISO	SYSFUN	Devuelve el valor de una indicación de la hora basado en un argumento de fecha, hora o indicación de la hora. Si el argumento es una fecha, inserta ceros para todos los elementos de hora. Si el argumento es una hora, inserta el valor de CURRENT DATE para los elementos de fecha y ceros para el elemento de fracción de hora.	
	DATE		TIMESTAMP
	TIME		TIMESTAMP
	TIMESTAMP		TIMESTAMP
	VARCHAR(26)		TIMESTAMP
TIMESTAMPDIFF	SYSFUN	Devuelve un número estimado de intervalos de tipo <i>argumento1</i> basado en la diferencia entre dos indicaciones de la hora. El segundo argumento es el resultado de restar los dos tipos de indicación de la hora y convertir el resultado a CHAR. Los valores válidos de intervalo (<i>argumento1</i>) son: 1 Fracciones de segundo 2 Segundos 4 Minutos 8 Horas 16 Días 32 Semanas 64 Meses 128 Trimestres 256 Años	
	INTEGER, CHAR(22)		INTEGER

Funciones

Tabla 15. Funciones soportadas (continuación)

Nombre de función	Esquema	Descripción
	Parámetros de entrada	
TRANSLATE	SYSIBM	Devuelve una serie en la que uno o más caracteres pueden haberse convertido a otros caracteres.
	CHAR	CHAR
	VARCHAR	VARCHAR
	CHAR, VARCHAR, VARCHAR	CHAR
	VARCHAR, VARCHAR, VARCHAR	VARCHAR
	CHAR, VARCHAR, VARCHAR, VARCHAR	CHAR
	VARCHAR, VARCHAR, VARCHAR, VARCHAR	VARCHAR
	GRAPHIC, VARGRAPHIC, VARGRAPHIC	GRAPHIC
	VARGRAPHIC, VARGRAPHIC, VARGRAPHIC	VARGRAPHIC
	GRAPHIC, VARGRAPHIC, VARGRAPHIC, VARGRAPHIC	GRAPHIC
	VARGRAPHIC, VARGRAPHIC, VARGRAPHIC, VARGRAPHIC	VARGRAPHIC
TRUNC o TRUNCATE	SYSFUN	Devuelve el <i>argumento1</i> truncado a <i>argumento2</i> posiciones a la derecha de la coma decimal. Si el <i>argumento2</i> es negativo, el <i>argumento1</i> se trunca al valor absoluto del <i>argumento2</i> posiciones a la izquierda de la coma decimal.
	INTEGER, INTEGER	INTEGER
	BIGINT, INTEGER	BIGINT
	DOUBLE, INTEGER	DOUBLE
TYPE_ID ³	SYSIBM	Devuelve el identificador de tipo de datos interno del tipo de datos dinámico del argumento. Tenga en cuenta que el resultado de esta función no es portable a través de las bases de datos.
	<i>cualquier-tipo-estructurado</i>	INTEGER
TYPE_NAME ³	SYSIBM	Devuelve el nombre no calificado del tipo de datos dinámico del argumento.
	<i>cualquier-tipo-estructurado</i>	VARCHAR(18)
TYPE_SCHEMA ³	SYSIBM	Devuelve el nombre de esquema del tipo dinámico del argumento.
	<i>cualquier-tipo-estructurado</i>	VARCHAR(128)
UCASE o UPPER	SYSIBM	Devuelve una serie en la que todos los caracteres se han convertido a mayúsculas.
	CHAR	CHAR
	VARCHAR	VARCHAR
UCASE	SYSFUN	Devuelve una serie en la que todos los caracteres se han convertido a mayúsculas.
	VARCHAR	VARCHAR
VALUE ³	SYSIBM	Igual que COALESCE.

Tabla 15. Funciones soportadas (continuación)

Nombre de función	Esquema	Descripción	
	Parámetros de entrada		Devuelve
VARCHAR	SYSIBM	Devuelve una representación VARCHAR del primer argumento. Si hay un segundo argumento presente, especifica la longitud del resultado.	
	<i>tipo-caracteres</i>		VARCHAR
	<i>tipo-caracteres</i> , INTEGER		VARCHAR
	<i>tipo-fechahora</i>		VARCHAR
VARGRAPHIC	SYSIBM	Devuelve una representación VARGRAPHIC del primer argumento. Si hay un segundo argumento presente, especifica la longitud del resultado.	
	<i>tipo-gráfico</i>		VARGRAPHIC
	<i>tipo-gráfico</i> , INTEGER		VARGRAPHIC
	VARCHAR		VARGRAPHIC
VARIANCE o VAR	SYSIBM	Devuelve la varianza de un conjunto de números (función de columna).	
	DOUBLE		DOUBLE
WEEK	SYSFUN	Devuelve la semana del año del argumento, en forma de valor entero comprendido dentro del rango 1-54.	
	VARCHAR(26)		INTEGER
	DATE		INTEGER
	TIMESTAMP		INTEGER
WEEK_ISO	SYSFUN	Devuelve la semana del año del argumento, en forma de valor entero comprendido dentro del rango 1-53. El primer día de la semana es lunes. La semana 1 es la primera semana del año que contendrá un jueves.	
	VARCHAR(26)		INTEGER
	DATE		INTEGER
	TIMESTAMP		INTEGER
YEAR	SYSIBM	Devuelve la parte correspondiente al año de un valor.	
	VARCHAR		INTEGER
	DATE		INTEGER
	TIMESTAMP		INTEGER
	DECIMAL		INTEGER
“+”	SYSIBM	Suma dos operandos numéricos.	
	<i>tipo-numérico</i> , <i>tipo-numérico</i>		<i>tipo-numérico máx</i>
“+”	SYSIBM	Operador más unitario.	
	<i>tipo-numérico</i>		<i>tipo-numérico</i>

Funciones

Tabla 15. Funciones soportadas (continuación)

Nombre de función	Esquema	Descripción	Devuelve
	Parámetros de entrada		
“+”	SYSIBM	Operador más de fecha y hora.	
	DATE, DECIMAL(8,0)		DATE
	TIME, DECIMAL(6,0)		TIME
	TIMESTAMP, DECIMAL(20,6)		TIMESTAMP
	DECIMAL(8,0), DATE		DATE
	DECIMAL(6,0), TIME		TIME
	DECIMAL(20,6), TIMESTAMP		TIMESTAMP
	<i>tipo-fechahora, DOUBLE, código-duración-etiquetada</i>		<i>tipo-fechahora</i>
“-”	SYSIBM	Resta dos operandos numéricos.	
	<i>tipo-numérico, tipo-numérico</i>		<i>tipo-numérico máx</i>
“_”	SYSIBM	Operador menos unitario.	
	<i>tipo-numérico</i>		<i>tipo-numérico</i> ¹
“_”	SYSIBM	Operador menos de fecha y hora.	
	DATE, DATE		DECIMAL(8,0)
	TIME, TIME		DECIMAL(6,0)
	TIMESTAMP, TIMESTAMP		DECIMAL(20,6)
	DATE, VARCHAR		DECIMAL(8,0)
	TIME, VARCHAR		DECIMAL(6,0)
	TIMESTAMP, VARCHAR		DECIMAL(20,6)
	VARCHAR, DATE		DECIMAL(8,0)
	VARCHAR, TIME		DECIMAL(6,0)
	VARCHAR, TIMESTAMP		DECIMAL(20,6)
	DATE, DECIMAL(8,0)		DATE
	TIME, DECIMAL(6,0)		TIME
	TIMESTAMP, DECIMAL(20,6)		TIMESTAMP
<i>tipo-fechahora, DOUBLE, código-duración-etiquetada</i>		<i>tipo-fechahora</i>	
“*”	SYSIBM	Multiplica dos operandos numéricos.	
	<i>tipo-numérico, tipo-numérico</i>		<i>tipo-numérico máx</i>
“/”	SYSIBM	Divide dos operadores numéricos.	
	<i>tipo-numérico, tipo-numérico</i>		<i>tipo-numérico máx</i>
“ ”	SYSIBM	Igual que CONCAT.	

Notas

- En las referencias a tipos de datos de serie que no se califican por una longitud, debe suponerse que dan soporte a la longitud máxima para el tipo de datos.
- En las referencias a un tipo de datos DECIMAL sin precisión ni escala, debe suponerse que se permite cualquier precisión y escala soportadas.

Claves para la tabla	
<i>cualquier-tipo-incorporado</i>	Cualquier tipo de datos que no sea un tipo diferenciado.
<i>cualquier-tipo</i>	Cualquier tipo definido en la base de datos.
<i>cualquier-tipo-estructurado</i>	Cualquier tipo estructurado definido por el usuario que esté definido para la base de datos.
<i>cualquier-tipo-comparable</i>	Cualquier tipo que sea comparable con otros tipos de argumentos, tal como está definido en “Asignaciones y comparaciones” en la página 105.
<i>cualquier-tipo-compatible-uni6n</i>	Cualquier tipo que sea compatible con otros tipos de argumentos, tal como está definido en “Reglas para los tipos de datos del resultado” en la página 120.
<i>tipo-caracteres</i>	Cualquier tipo de serie de caracteres: CHAR, VARCHAR, LONG VARCHAR, CLOB.
<i>tipo-serie-compatible</i>	Un tipo de serie que proviene de la misma agrupaci6n que otro argumento (p. ej., si un argumento es un <i>tipo-caracteres</i> , el otro tambi6n debe ser un <i>tipo-caracteres</i>).
<i>tipo-fechahora</i>	Cualquier otro tipo de indicaci6n de fecha y hora: DATE, TIME, TIMESTAMP.
<i>tipo-num6rico-exacto</i>	Cualquiera de los tipos num6ricos exactos: SMALLINT, INTEGER, BIGINT, DECIMAL
<i>tipo-gr6fico</i>	Cualquier tipo de serie de caracteres de doble byte: GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, DBCLOB.
<i>c6digo-duraci6n-etiquetada</i>	Como tipo es un SMALLINT. Si la funci6n se invoca utilizando la forma de infijo del operador m6s o menos, se pueden utilizar las duraciones-etiquetadas del apartado “Duraciones etiquetadas” en la p6gina 184. En una funci6n fuente que no utilice el car6cter de operador de m6s o menos como el nombre, se deben utilizar los siguientes valores para el argumento c6digo-duraci6n-etiquetada cuando se invoca la funci6n.
	1 YEAR o YEARS
	2 MONTH o MONTHS
	3 DAY o DAYS
	4 HOUR o HOURS
	5 MINUTE o MINUTES
	6 SECOND o SECONDS
	7 MICROSECOND o MICROSECONDS
<i>tipo-LOB</i>	Cualquier tipo de gran objeto: BLOB, CLOB, DBCLOB.
<i>tipo-num6rico-m6x</i>	El tipo num6rico m6ximo de los argumentos donde m6ximo se define como el <i>tipo-num6rico</i> m6s a la derecha.
<i>tipo-serie-m6x</i>	El tipo de serie m6ximo de los argumentos donde m6ximo se define como el <i>tipo-caracteres</i> o el <i>tipo-gr6fico</i> m6s a la derecha. Si los argumentos son BLOB, el <i>tipo-serie-m6x</i> es BLOB.
<i>tipo-num6rico</i>	Cualquiera de los tipos num6ricos: SMALLINT, INTEGER, BIGINT, DECIMAL, REAL, DOUBLE.
<i>tipo-serie</i>	Cualquier tipo de <i>tipo de caracteres</i> , <i>tipo-gr6fico</i> o BLOB.

Funciones

Notas de la tabla

- 1 Cuando el parámetro de entrada es SMALLINT, el tipo del resultado es INTEGER. Cuando el parámetro de entrada es REAL, el tipo del resultado es DOUBLE.
- 2 Las palabras clave permitidas son ISO, USA, EUR, JIS y LOCAL. Esta signatura de función no está soportada como función derivada.
- 3 Esta función no se puede utilizar como función fuente.
- 4 Las palabras clave ALL o DISTINCT pueden utilizarse antes del primer parámetro. Si se especifica DISTINCT, no se da soporte a la utilización de tipos estructurados definidos por el usuario, tipos de series largas ni de un tipo DATALINK.
- 5 No se da soporte a la utilización de tipos estructurados definidos por el usuario, tipos de series largas ni de un tipo DATALINK.
- 6 El tipo devuelto por RAISE_ERROR depende del contexto de su utilización. RAISE_ERROR, si no hay conversión hacia un tipo determinado, devolverá un tipo adecuado para su invocación en una expresión CASE.
- 7 No se soporta el uso de *tipo-gráfico*, *tipo-LOB*, tipos de serie larga y tipos DATALINK.

Funciones de columna

El argumento de una función de columna es un conjunto de valores derivados de una expresión. La expresión puede incluir columnas pero no puede incluir una *selección completa-escalar* ni otra función de columna (SQLSTATE 42607). El ámbito del conjunto es un grupo o una tabla resultante intermedia tal como se explica en el “Capítulo 5. Consultas” en la página 471.

Si se especifica una cláusula GROUP BY en una consulta y el resultado intermedio de las cláusulas FROM, WHERE, GROUP BY y HAVING es un conjunto vacío; las funciones de columna no se aplican, el resultado de la consulta es el conjunto vacío, el SQLCODE se establece en +100 y el SQLSTATE se establece en '02000'.

Si no se especifica ninguna cláusula GROUP BY en una consulta y el resultado intermedio de las cláusulas FROM, WHERE y HAVING es el conjunto vacío, se aplican las funciones de columna al conjunto vacío.

Por ejemplo, el resultado de la siguiente sentencia SELECT es el número de valores diferenciado de JOBCODE para los empleados en el departamento D01:

```
SELECT COUNT(DISTINCT JOBCODE)
FROM CORPDATA.EMPLOYEE
WHERE WORKDEPT = 'D01'
```

La palabra clave DISTINCT no se considera un argumento de la función, sino una especificación de una operación que se realiza antes de aplicar la función. Si se especifica DISTINCT, se eliminan los valores duplicados. Si se especifica ALL implícita o explícitamente, no se eliminan los valores duplicados.

Se pueden utilizar expresiones en las funciones de columna, por ejemplo:

```
SELECT MAX(BONUS + 1000)
INTO :TOP_SALESREP_BONUS
FROM EMPLOYEE
WHERE COMM > 5000
```

Las funciones de columna que siguen están en el esquema SYSIBM y pueden calificarse con el nombre de esquema (por ejemplo, SYSIBM.COUNT(*)).

AVG

AVG



El esquema es SYSIBM.

La función AVG devuelve el promedio de un conjunto de números.

Los valores del argumento deben ser números (sólo tipos incorporados) y su suma debe estar dentro del rango del tipo de datos del resultado, excepto para un tipo de datos de resultado decimal. Para los resultados decimales, la suma debe estar dentro del rango soportado por un tipo de datos decimal que tenga una precisión de 31 y una escala idéntica a la escala de los valores del argumento. El resultado puede ser nulo.

El tipo de datos del resultado es el mismo que el tipo de datos de los valores del argumento, excepto que:

- El resultado es un entero grande si los valores del argumento son enteros pequeños.
- El resultado es de coma flotante de precisión doble si los valores del argumento son de coma flotante de precisión simple.

Si el tipo de datos de los valores del argumento es decimal con la precisión p y la escala s , la precisión del resultado es 31 y la escala es $31-p+s$.

La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos. Si se especifica DISTINCT, se eliminan los valores duplicados redundantes.

Si la función se aplica a un conjunto vacío, el resultado es un valor nulo. De lo contrario, el resultado es el valor promedio del conjunto.

El orden en el que los valores se añaden es indefinido, pero cada resultado intermedio debe estar en el rango del tipo de datos del resultado.

Si el tipo del resultado es entero, se pierde la parte correspondiente a la fracción del promedio.

Ejemplos:

- Utilizando la tabla PROJECT, establezca la variable del lenguaje principal AVERAGE (decimal(5,2)) en el nivel promedio de los trabajadores (PRSTAFF) de los proyectos del departamento (DEPTNO) 'D11'.


```

SELECT AVG(PRSTAFF)
  INTO :AVERAGE
  FROM PROJECT
  WHERE DEPTNO = 'D11'

```

Da como resultado que AVERAGE se establece en 4,25 (es decir 17/4) cuando se utiliza la tabla de ejemplo.

- Utilizando la tabla PROJECT, establezca la variable del lenguaje principal ANY_CALC (decimal(5,2)) en el promedio de cada valor de nivel exclusivo de los trabajadores (PRSTAFF) de proyectos del departamento (DEPTNO) 'D11'.

```

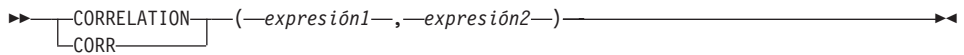
SELECT AVG(DISTINCT PRSTAFF)
  INTO :ANY_CALC
  FROM PROJECT
  WHERE DEPTNO = 'D11'

```

El resultado es que ANY_CALC se establece en 4,66 (es decir 14/3) cuando se utiliza la tabla de ejemplo.

CORRELATION

CORRELATION



El esquema es SYSIBM.

La función CORRELATION devuelve el coeficiente de correlación de un conjunto de pares de números.

Los valores del argumento deben ser números.

El tipo de datos del resultado es de coma flotante de precisión doble. El resultado puede ser nulo. Cuando no es nulo, el resultado está entre -1 y 1.

La función se aplica al conjunto de pares (*expresión1*, *expresión2*) derivado de los valores del argumento por la eliminación de todos los pares para los que *expresión1* o *expresión2* es nulo.

Si la función se aplica a un conjunto vacío o si $STDDEV(\textit{expresión1})$ o $STDDEV(\textit{expresión2})$ es igual a cero, el resultado es un valor nulo. De lo contrario, el resultado es el coeficiente de correlación para los pares de valores del conjunto. El resultado es equivalente a la expresión siguiente:

$$COVARIANCE(\textit{expresión1}, \textit{expresión2}) / (STDDEV(\textit{expresión1}) * STDDEV(\textit{expresión2}))$$

El orden en el que los valores se agregan no está definido, pero cada resultado intermedio debe estar dentro del rango del tipo de datos del resultado.

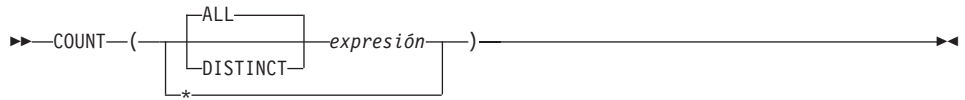
Ejemplo:

- Utilizando la tabla EMPLOYEE, establezca la variable de lenguaje principal CORRLN (coma flotante de precisión doble) en la correlación entre salario y bono para los empleados del departamento (WORKDEPT) 'A00'.

```
SELECT CORRELATION(SALARY, BONUS)
      INTO :CORRLN
      FROM EMPLOYEE
      WHERE WORKDEPT = 'A00'
```

CORRLN se establece en 9,99853953399538E-001 aproximadamente cuando se utiliza la tabla de ejemplo.

COUNT



El esquema es SYSIBM.

La función COUNT devuelve el número de filas o valores de un conjunto de filas o valores.

Si se utiliza DISTINCT, el tipo de datos resultante de *expresión* no debe tener una longitud mayor que 255 para una columna de caracteres, o que 127 para una columna gráfica. El tipo de datos de *expresión* no puede ser un LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB, DATALINK, un tipo diferenciado para cualquiera de estos tipos ni un tipo estructurado (SQLSTATE 42907).

El resultado de la función es un entero grande. El resultado no puede ser nulo.

El argumento de COUNT(*) es un conjunto de filas. El resultado es el número de filas del conjunto. Una fila que sólo incluye valores NULL se incluye en la cuenta.

El argumento de COUNT(DISTINCT *expresión*) es un conjunto de valores. La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos y duplicados. El resultado es el número de distintos valores no nulos del conjunto.

El argumento de COUNT(*expresión*) o COUNT(ALL *expresión*) es un conjunto de valores. La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos. El resultado es el número de valores no nulos del conjunto, incluyendo los duplicados.

Ejemplos:

- Utilizando la tabla EMPLOYEE, establezca la variable del lenguaje principal FEMALE (int) en el número de filas en que el valor de la columna SEX es 'F'.

```
SELECT COUNT(*)
      INTO :FEMALE
      FROM EMPLOYEE
      WHERE SEX = 'F'
```

COUNT

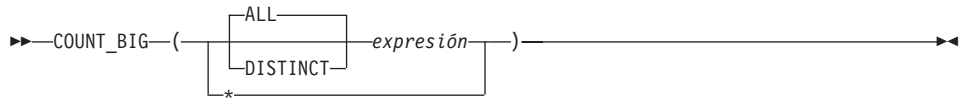
El resultado es que FEMALE se establece en 13 cuando se utiliza la tabla de ejemplo.

- Utilizando la tabla EMPLOYEE, establezca la variable del lenguaje principal FEMALE_IN_DEPT (int) en el número de departamentos (WORKDEPT) que tienen como mínimo una mujer como miembro.

```
SELECT COUNT(DISTINCT WORKDEPT)
INTO :FEMALE_IN_DEPT
FROM EMPLOYEE
WHERE SEX = 'F'
```

El resultado es que FEMALE_IN_DEPT se establece en 5 cuando se utiliza la tabla de ejemplo. (Hay como mínimo una mujer en los departamentos A00, C01, D11, D21 y E11.)

COUNT_BIG



El esquema es SYSIBM.

La función COUNT_BIG devuelve el número de filas o valores de un conjunto de filas o valores. Es similar a COUNT excepto que el resultado puede ser mayor que el valor máximo de entero.

Si se utiliza DISTINCT, el tipo de datos resultante de *expresión* no debe tener una longitud mayor que 255 para una columna de caracteres, o que 127 para una columna gráfica. El tipo de datos resultante de *expresión* no puede ser un LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB, DATALINK, un tipo diferenciado para cualquiera de estos tipos ni un tipo estructurado (SQLSTATE 42907).

El resultado de la función es un decimal con la precisión 31 y la escala 0. El resultado no puede ser nulo.

El argumento de COUNT_BIG(*) es un conjunto de filas. El resultado es el número de filas del conjunto. Una fila que sólo incluye valores NULL se incluye en la cuenta.

El argumento de COUNT_BIG(DISTINCT *expresión*) es un conjunto de valores. La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos y duplicados. El resultado es el número de distintos valores no nulos del conjunto.

El argumento de COUNT_BIG(*expresión*) o COUNT_BIG(ALL *expresión*) es un conjunto de valores. La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos. El resultado es el número de valores no nulos del conjunto, incluyendo los duplicados.

Ejemplos:

- Consulte los ejemplos de COUNT y sustituya COUNT_BIG por las apariciones de COUNT. Los resultados son los mismos excepto por el tipo de datos del resultado.
- Algunas aplicaciones pueden necesitar la utilización de COUNT pero necesitan dar soporte a valores mayores que el entero más grande. Esto se puede conseguir mediante la utilización de funciones derivadas definidas por el usuario y la definición de la vía de acceso de SQL. Las siguientes

COUNT_BIG

series de sentencias muestran cómo crear una función derivada para dar soporte a COUNT(*) basándose en COUNT_BIG y devolver un valor decimal con una precisión de 15. La vía de acceso de SQL se establece de manera que se utilice la función derivada basada en COUNT_BIG en las sentencias subsiguientes, tal como la consulta mostrada.

```
CREATE FUNCTION RICK.COUNT() RETURNS DECIMAL(15,0)  
    SOURCE SYSIBM.COUNT_BIG();  
SET CURRENT FUNCTION PATH RICK, SYSTEM PATH;  
SELECT COUNT(*) FROM EMPLOYEE;
```

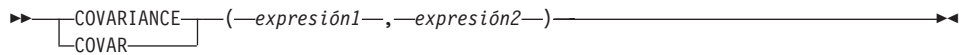
Observe que la función derivada se define sin parámetros para dar soporte a COUNT(*). Esto sólo es efectivo si utiliza COUNT como nombre de la función y no califica la función con el nombre de esquema cuando se utiliza. Para conseguir el mismo efecto que COUNT(*) con un nombre distinto de COUNT, invoque la función sin parámetros. Por lo tanto, si RICK.COUNT se ha definido como RICK.MYCOUNT, la consulta se tendría que haber escrito de la siguiente manera:

```
SELECT MYCOUNT() FROM EMPLOYEE;
```

Si la cuenta se efectúa en una columna específica, la función derivada debe especificar el tipo de columna. Las sentencias siguientes crean una función derivada que tomará cualquier columna CHAR como argumento y utilizará COUNT_BIG para realizar el conteo.

```
CREATE FUNCTION RICK.COUNT(CHAR()) RETURNS DOUBLE  
    SOURCE SYSIBM.COUNT_BIG(CHAR());  
SELECT COUNT(DISTINCT WORKDEPT) FROM EMPLOYEE;
```

COVARIANCE



El esquema es SYSIBM.

La función COVARIANCE devuelve la covarianza (del contenido) de un conjunto de pares de números.

Los valores del argumento deben ser números.

El tipo de datos del resultado es de coma flotante de precisión doble. El resultado puede ser nulo.

La función se aplica al conjunto de pares (*expresión1*, *expresión2*) derivado de los valores del argumento por la eliminación de todos los pares para los que *expresión1* o *expresión2* es nulo.

Si la función se aplica a un conjunto vacío, el resultado es un valor nulo. De lo contrario, el resultado es la covarianza de los pares de valores del conjunto. El resultado es equivalente a lo siguiente:

1. Establezca que avgexp1 es el resultado de $AVG(\text{expresión1})$ y que avgexp2 es el resultado de $AVG(\text{expresión2})$.
2. El resultado de $COVARIANCE(\text{expresión1}, \text{expresión2})$ es $AVG((\text{expresión1} - \text{avgexp1}) * (\text{expresión2} - \text{avgexp2}))$

El orden en el que los valores se agregan no está definido, pero cada resultado intermedio debe estar dentro del rango del tipo de datos del resultado.

Ejemplo:

- Utilizando la tabla EMPLOYEE, establezca la variable del lenguaje principal COVARNCE (coma flotante de precisión doble) en la covarianza entre salario y bono para los empleados del departamento (WORKDEPT) 'A00'.

```
SELECT COVARIANCE(SALARY, BONUS)
      INTO :COVARNCE
      FROM EMPLOYEE
      WHERE WORKDEPT = 'A00'
```

COVARNCE se establece en 1,68888888888889E+006 aproximadamente cuando se utiliza la tabla de ejemplo.

GROUPING

GROUPING

►—GROUPING—(—*expresión*—)————►

El esquema es SYSIBM.

Utilizada con conjuntos-agrupaciones y supergrupos (consulte el apartado “Cláusula group-by” en la página 487 para ver los detalles), la función GROUPING devuelve un valor que indica si una fila devuelta en un conjunto de respuestas de GROUP BY es una fila generada por un conjunto de agrupaciones que excluye la columna representada por la *expresión* o no.

El argumento puede ser de cualquier tipo, pero debe ser un elemento de una cláusula GROUP BY.

El resultado de la función es un entero pequeño. Se establece en uno de los valores siguientes:

- 1 El valor de la *expresión* de la fila devuelta es un valor nulo y la fila se ha generado por el supergrupo. Esta fila generada puede utilizarse para proporcionar valores de subtotales para la expresión GROUP BY.
- 0 El valor no es el de arriba.

Ejemplo:

La siguiente consulta:

```
SELECT SALES_DATE,  
       SALES_PERSON,  
       SUM(SALES) AS UNITS_SOLD,  
       GROUPING(SALES_DATE) AS DATE_GROUP,  
       GROUPING(SALES_PERSON) AS SALES_GROUP  
FROM SALES  
GROUP BY CUBE (SALES_DATE, SALES_PERSON)  
ORDER BY SALES_DATE, SALES_PERSON
```

da como resultado:

SALES_DATE	SALES_PERSON	UNITS_SOLD	DATE_GROUP	SALES_GROUP
12/31/1995	GOUNOT	1	0	0
12/31/1995	LEE	6	0	0
12/31/1995	LUCCHESSI	1	0	0
12/31/1995	-	8	0	1
03/29/1996	GOUNOT	11	0	0
03/29/1996	LEE	12	0	0
03/29/1996	LUCCHESSI	4	0	0
03/29/1996	-	27	0	1

GROUPING

03/30/1996	GOUNOT	21	0	0
03/30/1996	LEE	21	0	0
03/30/1996	LUCCHESSI	4	0	0
03/30/1996	-	46	0	1
03/31/1996	GOUNOT	3	0	0
03/31/1996	LEE	27	0	0
03/31/1996	LUCCHESSI	1	0	0
03/31/1996	-	31	0	1
04/01/1996	GOUNOT	14	0	0
04/01/1996	LEE	25	0	0
04/01/1996	LUCCHESSI	4	0	0
04/01/1996	-	43	0	1
-	GOUNOT	50	1	0
-	LEE	91	1	0
-	LUCCHESSI	14	1	0
-	-	155	1	1

Una aplicación puede reconocer una fila de subtotales de SALES_DATE por el hecho de que el valor de DATE_GROUP es 0 y el valor de SALES_GROUP es 1. Una fila de subtotales SALES_PERSON puede reconocerse por el hecho de que el valor de DATE_GROUP es 1 y el valor de SALES_GROUP es 0. Un fila de total general puede reconocerse por el valor 1 de DATE_GROUP y SALES_GROUP.

MAX

MAX



El esquema es SYSIBM.

La función MAX devuelve el valor máximo de un conjunto de valores.

Los valores del argumento pueden ser de cualquier tipo incorporado que no sea una serie larga ni DATALINK.

Si se utiliza DISTINCT, el tipo de datos resultante de *expresión* no debe tener una longitud mayor que 255 para una columna de caracteres, o que 127 para una columna gráfica. El tipo de datos resultante de *expresión* no puede ser un LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB, DATALINK, un tipo diferenciado para cualquiera de estos tipos ni un tipo estructurado (SQLSTATE 42907).

El tipo de datos, la longitud y la página de códigos del resultado son iguales que el tipo de datos, la longitud y la página de códigos de los valores del argumento. El resultado se considera un valor derivado y puede ser nulo.

La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos.

Si la función se aplica a un conjunto vacío, el resultado es un valor nulo. De lo contrario, el resultado es el valor máximo del conjunto.

La especificación de DISTINCT no tiene ningún efecto en el resultado y, por lo tanto, no es aconsejable. Se incluye para la compatibilidad con otros sistemas relacionados.

Ejemplos:

- Utilizando la tabla EMPLOYEE, establezca la variable del lenguaje principal MAX_SALARY (decimal(7,2)) en el valor del salario máximo mensual (SALARY/12).

```
SELECT MAX(SALARY) / 12
      INTO :MAX_SALARY
      FROM EMPLOYEE
```

El resultado es que MAX_SALARY se establece en 4395,83 cuando se utiliza esta tabla de ejemplo.

- Utilizando la tabla PROJECT, establezca la variable del lenguaje principal LAST_PROJ(char(24)) en el nombre de proyecto (PROJNAME) que es el último en el orden de clasificación.

```
SELECT MAX(PROJNAME)  
INTO :LAST_PROJ  
FROM PROJECT
```

Da como resultado que LAST_PROJ se establece en 'WELD LINE PLANNING' cuando se utiliza la tabla de ejemplo.

- De manera parecida al ejemplo anterior, establezca la variable del lenguaje principal LAST_PROJ (char(40)) en el nombre del proyecto que es el último en el orden de clasificación cuando se concatena un nombre de proyecto con la variable del lenguaje principal PROJSUPP. PROJSUPP es '_Support'; tiene un tipo de datos char(8).

```
SELECT MAX(PROJNAME CONCAT PROJSUPP)  
INTO :LAST_PROJ  
FROM PROJECT
```

Da como resultado que LAST_PROJ se establece en 'WELD LINE PLANNING_SUPPORT' cuando se utiliza la tabla de ejemplo.

MIN

MIN



El esquema es SYSIBM.

La función MIN devuelve el valor mínimo de un conjunto de valores.

Los valores del argumento pueden ser de cualquier tipo incorporado que no sea una serie larga ni DATALINK.

Si se utiliza DISTINCT, el tipo de datos resultante de *expresión* no debe tener una longitud mayor que 255 para una columna de caracteres, o que 127 para una columna gráfica. El tipo de datos resultante de *expresión* no puede ser un LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB, DATALINK, un tipo diferenciado para cualquiera de estos tipos ni un tipo estructurado (SQLSTATE 42907).

El tipo de datos, la longitud y la página de códigos del resultado son iguales que el tipo de datos, la longitud y la página de códigos de los valores del argumento. El resultado se considera un valor derivado y puede ser nulo.

La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos.

Si la función se aplica a un conjunto vacío, el resultado de la función es un valor nulo. De lo contrario, el resultado es el valor mínimo del conjunto.

La especificación de DISTINCT no tiene ningún efecto en el resultado y, por lo tanto, no es aconsejable. Se incluye para la compatibilidad con otros sistemas relacionados.

Ejemplos:

- Utilizando la tabla EMPLOYEE, establezca la variable del lenguaje principal COMM_SPREAD (decimal(7,2)) en la diferencia entre la comisión máxima y mínima (COMM) de los miembros del departamento (WORKDEPT) 'D11'.

```
SELECT MAX(COMM) - MIN(COMM)
      INTO :COMM_SPREAD
      FROM EMPLOYEE
      WHERE WORKDEPT = 'D11'
```

El resultado es que COMM_SPREAD se establece en 1118 (es decir, 2580 - 1462) cuando se utiliza la tabla de ejemplo.

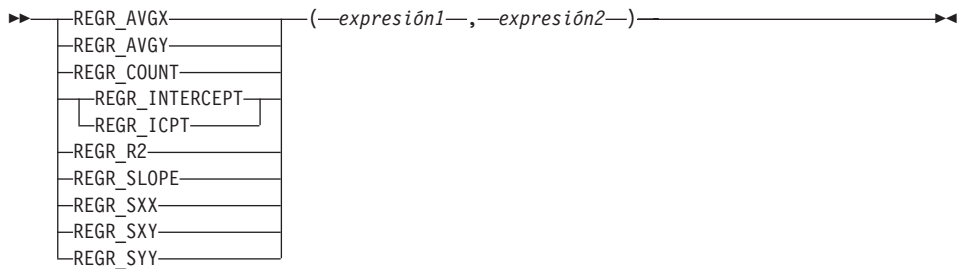
- Utilizando la tabla PROJECT, establezca la variable del lenguaje principal (FIRST_FINISHED (char(10)) en la fecha de finalización estimada (PRENDATE) del primer proyecto que se ha de terminar.

```
SELECT MIN(PRENDATE)  
INTO :FIRST_FINISHED  
FROM PROJECT
```

Da como resultado que FIRST_FINISHED se establece en '1982-09-15' cuando se utiliza la tabla de ejemplo.

Funciones REGRESSION

Funciones REGRESSION



El esquema es SYSIBM.

Las funciones de regresión soportan la adecuación de una línea de regresión mínimo-cuadrados-normales del formato $y = a * x + b$ para un conjunto de pares de números. El primer elemento de cada par (*expresión1*) se interpreta como un valor de la variable dependiente (p. ej., un "valor y"). El segundo elemento de cada par (*expresión2*) se interpreta como un valor de la variable independiente (p. ej., un "valor x").

La función REGR_COUNT devuelve el número de pares de números no nulos utilizados para acomodar la línea de regresión (vea más abajo).

La función REGR_INTERCEPT (su formato corto es REGR_ICPT) devuelve la intersección y de la línea de regresión ("b" en la ecuación anterior)

La función REGR_R2 devuelve el coeficiente de determinación (llamado también "cuadrado-R" o "mejor-adecuación") para la regresión.

La función REGR_SLOPE devuelve la inclinación de la línea (el parámetro "a" de la ecuación anterior).

Las funciones REGR_AVGX, REGR_AVGY, REGR_SXX, REGR_SYY y REGR_SXY devuelven cantidades que pueden utilizarse para calcular varias estadísticas de diagnóstico necesarias para la evaluación de la calidad y la validez estadística del modelo de regresión (vea más abajo).

Los valores del argumento deben ser números.

El tipo de datos del resultado de REGR_COUNT es un entero. Para las demás funciones, el tipo de datos del resultado es de coma flotante de precisión doble. El resultado puede ser nulo. Cuando no es nulo, el resultado de REGR_R2 está entre 0 y 1 y el resultado de REGR_SXX y REGR_SYY no es negativo.

Cada función se aplica al conjunto de pares (*expresión1*, *expresión2*) derivado de los valores del argumento por la eliminación de todos los pares para los que *expresión1* o *expresión2* es nulo.

Si el conjunto no está vacío y $VARIANCE(\textit{expresión2})$ es positivo, $REGR_COUNT$ devuelve el número de pares no nulos del conjunto y las demás funciones devuelven los resultados que se definen de la siguiente manera:

```

REGR_SLOPE(expresión1,expresión2) =
    COVARIANCE(expresión1,expresión2)/VARIANCE(expresión2)
REGR_INTERCEPT(expresión1, expresión2) =
    AVG(expresión1) - REGR_SLOPE(expresión1, expresión2) * AVG(expresión2)
REGR_R2(expresión1, expresión2) =
    POWER(CORRELATION(expresión1, expresión2), 2) if VARIANCE(expresión1)>0
    REGR_R2(expresión1, expresión2) = 1 if VARIANCE(expresión1)=0
REGR_AVGX(expresión1, expresión2) = AVG(expresión2)
REGR_AVGY(expresión1, expresión2) = AVG(expresión1)
REGR_SXX(expresión1, expresión2) =
    REGR_COUNT(expresión1, expresión2) * VARIANCE(expresión2)
REGR_SYY(expresión1, expresión2) =
    REGR_COUNT(expresión1, expresión2) * VARIANCE(expresión1)
REGR_SXY(expresión1, expresión2) =
    REGR_COUNT(expresión1, expresión2) * COVARIANCE(expresión1, expresión2)
    
```

Si el conjunto no está vacío y $VARIANCE(\textit{expresión2})$ es igual a cero, la línea de regresión tiene una inclinación infinita o no está definida. En este caso, las funciones $REGR_SLOPE$, $REGR_INTERCEPT$ y $REGR_R2$ devuelven cada una un valor nulo y las demás funciones devuelven valores tal como se ha definido arriba. Si el conjunto está vacío, $REGR_COUNT$ devuelve cero y las demás funciones devuelven un valor nulo.

El orden en el que los valores se agregan no está definido, pero cada resultado intermedio debe estar dentro del rango del tipo de datos del resultado.

Las funciones de regresión se calculan simultáneamente durante un solo paso a través de los datos. En general, es más eficaz utilizar las funciones de regresión para calcular las estadísticas necesarias para un análisis de regresión que realizar cálculos equivalentes utilizando las funciones normales de columna como $AVERAGE$, $VARIANCE$, $COVARIANCE$, etcétera.

Las estadísticas de diagnóstico normales que acompañan a un análisis de regresión-lineal se pueden calcular en términos de las funciones anteriores. Por ejemplo:

R2 ajustada

$$1 - ((1 - REGR_R2) * ((REGR_COUNT - 1) / (REGR_COUNT - 2)))$$

Funciones REGRESSION

Error estándar

$$\text{SQRT} \left(\frac{\text{REGR_SYY} - (\text{POWER}(\text{REGR_SXY}, 2) / \text{REGR_SXX})}{\text{REGR_COUNT} - 2} \right)$$

Suma total de cuadrados

$$\text{REGR_SYY}$$

Suma de cuadrados de regresión

$$\text{POWER}(\text{REGR_SXY}, 2) / \text{REGR_SXX}$$

Suma de cuadrados residuales

$$(\text{Suma total de cuadrados}) - (\text{Suma de cuadrados de regresión})$$

t estadística de inclinación

$$\text{REGR_SLOPE} * \text{SQRT}(\text{REGR_SXX}) / (\text{Error estándar})$$

t estadística para intersección y

$$\frac{\text{REGR_INTERCEPT}}{(\text{Error estándar}) * \text{SQRT} \left(\frac{1}{\text{REGR_COUNT}} + \frac{\text{POWER}(\text{REGR_AVGX}, 2)}{\text{REGR_SXX}} \right)}$$

Ejemplo:

- Utilizando la tabla EMPLOYEE, calcule la línea de regresión de cuadrados-mínimos-normales que expresa el bono de un empleado del departamento (WORKDEPT) 'A00' como una función lineal del salario del empleado. Establezca las variables del lenguaje principal SLOPE, ICPT, RSQR (coma flotante de precisión doble) en la inclinación, intersección y coeficiente de determinación de la línea de regresión, respectivamente. Establezca también las variables del lenguaje principal AVGSAL y AVGBONUS en el salario promedio y el bono promedio, respectivamente, de los empleados del departamento 'A00', y establezca la variable del lenguaje principal CNT (entero) en el número de empleados del departamento 'A00' para los que están disponibles los datos de salario y de bono. Almacene las demás estadísticas de regresión en las variables del lenguaje principal SXX, SYY y SXY.

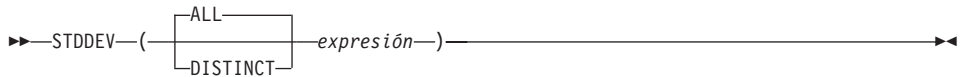
```
SELECT REGR_SLOPE(BONUS, SALARY), REGR_INTERCEPT(BONUS, SALARY),  
       REGR_R2(BONUS, SALARY), REGR_COUNT(BONUS, SALARY),  
       REGR_AVGX(BONUS, SALARY), REGR_AVGY(BONUS, SALARY),  
       REGR_SXX(BONUS, SALARY), REGR_SYY(BONUS, SALARY),  
       REGR_SXY(BONUS, SALARY)  
INTO :SLOPE, :ICPT,  
     :RSQR, :CNT,  
     :AVGSAL, :AVGBONUS,  
     :SXX, :SYY,  
     :SXY  
FROM EMPLOYEE  
WHERE WORKDEPT = 'A00'
```

Al utilizar la tabla de ejemplo, las variables del lenguaje principal se establecen en los siguientes valores aproximados:

SLOPE: +1,71002671916749E-002
ICPT: +1,00871888623260E+002
RSQR: +9,99707928128685E-001
CNT: 3
AVGSAL: +4,28333333333333E+004
AVGBONUS: +8,33333333333333E+002
SXX: +2,96291666666667E+008
SYY: +8,66666666666667E+004
SXY: +5,06666666666667E+006

STDDEV

STDDEV



El esquema es SYSIBM.

La función STDDEV devuelve la desviación estándar de un conjunto de números.

Los valores del argumento deben ser números.

El tipo de datos del resultado es de coma flotante de precisión doble. El resultado puede ser nulo.

La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos. Si se especifica DISTINCT, se eliminan los valores duplicados redundantes.

Si la función se aplica a un conjunto vacío, el resultado es un valor nulo. De lo contrario, el resultado es la desviación estándar de los valores del conjunto.

El orden en el que los valores se agregan no está definido, pero cada resultado intermedio debe estar dentro del rango del tipo de datos resultante.

Ejemplo:

- Utilizando la tabla EMPLOYEE, establezca la variable del lenguaje principal DEV (coma flotante de precisión doble) en la desviación estándar de los salarios para los empleados del departamento (WORKDEPT) 'A00'.

```
SELECT STDDEV(SALARY)
      INTO :DEV
      FROM EMPLOYEE
      WHERE WORKDEPT = 'A00'
```

Da como resultado que DEV se establece en 9938,00 aproximadamente cuando se utiliza la tabla de ejemplo.

SUM



El esquema es SYSIBM.

La función SUM devuelve la suma de un conjunto de números.

Los valores del argumento deben ser números (sólo tipos incorporados) y su suma debe estar dentro del rango del tipo de datos del resultado.

El tipo de datos del resultado es el mismo que el tipo de datos de los valores del argumento excepto que:

- El resultado es un entero grande si los valores del argumento son enteros pequeños.
- El resultado es de coma flotante de precisión doble si los valores del argumento son de coma flotante de precisión simple.

Si el tipo de datos de los valores del argumento es decimal, la precisión del resultado es 31 y la escala es la misma que la de los valores del argumento. El resultado puede ser nulo.

La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos. Si se especifica DISTINCT, también se eliminan los valores duplicados redundantes.

Si la función se aplica a un conjunto vacío, el resultado es un valor nulo. De lo contrario, el resultado es la suma de los valores del conjunto.

Ejemplo:

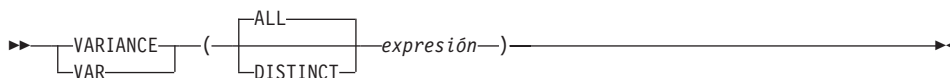
- Utilizando la tabla EMPLOYEE, establezca la variable del lenguaje principal JOB_BONUS (decimal(9,2)) en el total de bonificaciones (BONUS) pagadas a los conserjes (JOB='CLERK').

```
SELECT SUM(BONUS)
      INTO :JOB_BONUS
      FROM EMPLOYEE
      WHERE JOB = 'CLERK'
```

El resultado es que JOB_BONUS se establece en 2800 cuando se utiliza la tabla de ejemplo.

VARIANCE

VARIANCE



El esquema es SYSIBM.

La función VARIANCE devuelve la varianza de un conjunto de números.

Los valores del argumento deben ser números.

El tipo de datos del resultado es de coma flotante de precisión doble. El resultado puede ser nulo.

La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos. Si se especifica DISTINCT, se eliminan los valores duplicados redundantes.

Si la función se aplica a un conjunto vacío, el resultado es un valor nulo. De lo contrario, el resultado es la varianza de los valores del conjunto.

El orden en el que los valores se añaden es indefinido, pero cada resultado intermedio debe estar en el rango del tipo de datos del resultado.

Ejemplo:

- Utilizando la tabla EMPLOYEE, establezca la variable del lenguaje principal VARNCE (coma flotante de precisión doble) en la varianza de los salarios para los empleados del departamento (WORKDEPT) 'A00'.

```
SELECT VARIANCE(SALARY)
INTO :VARNCE
FROM EMPLOYEE
WHERE WORKDEPT = 'A00'
```

Da como resultado que VARNCE se establece en 98763888,88 aproximadamente cuando se utiliza la tabla de ejemplo.

Funciones escalares

Una función escalar se puede utilizar siempre que se pueda utilizar una expresión. Sin embargo, las restricciones que se aplican a la utilización de expresiones y a las funciones de columna también se aplican cuando se utiliza una expresión o una función de columna en una función escalar. Por ejemplo, el argumento de una función escalar sólo puede ser una función de columna si está permitida una función de columna en el contexto en el que se utiliza la función escalar.

Las restricciones a la utilización de funciones de columna no se aplican a las funciones escalares porque una función escalar se aplica a un solo valor en lugar de a un conjunto de valores.

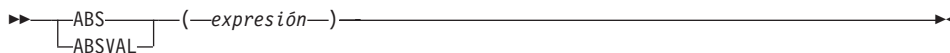
Ejemplo: El resultado de la siguiente sentencia SELECT contiene el mismo número de filas como el número de empleados que hay en el departamento D01:

```
SELECT EMPNO, LASTNAME, YEAR(CURRENT DATE - BRTHDATE)
      FROM EMPLOYEE
 WHERE WORKDEPT = 'D01'
```

Las siguientes funciones escalares pueden calificarse con el nombre de esquema (por ejemplo, SYSIBM.CHAR(123)).

ABS o ABSVAL

ABS o ABSVAL



El esquema es SYSIBM.

Esta función está disponible por primera vez en el FixPak 2 de la Versión 7.1.

Nota: La versión SYSFUN de la función ABS (o ABSVAL) continúa estando disponible.

Devuelve el valor absoluto del argumento.

El argumento es una expresión que devuelve un valor de cualquier tipo de datos numérico incorporado.

El resultado de la función tiene el mismo tipo de datos y el mismo atributo de longitud que el argumento. Si el argumento puede ser nulo o la base de datos está configurada con DFT_SQLMATHWARN establecido en Sí, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Por ejemplo:

ABS(-51234)

devuelve un INTEGER con un valor de 51234.

ACOS

►►—ACOS—(*—expresión—*)—►►

El esquema es SYSFUN.

Devuelve el coseno del arco del argumento como un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo de datos incorporado. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

ASCII

ASCII

►►ASCII(—*expresión*—)◄◄

El esquema es SYSFUN.

Devuelve el valor de código ASCII del carácter situado más a la izquierda del argumento como un entero.

El argumento puede ser de cualquier tipo de serie de caracteres incorporado. Para un VARCHAR, la longitud máxima es 4.000 bytes y para un CLOB la longitud máxima es 1.048.576 bytes. LONG VARCHAR se convierte a CLOB para que lo procese la función.

El resultado de la función siempre es INTEGER.

El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

ASIN

►► ASIN(*—expresión—*) ◀◀

El esquema es SYSFUN.

Devuelve el seno del arco del argumento como un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo numérico incorporado. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

ATAN

ATAN

►► ATAN(*—expresión—*) ◀◀

El esquema es SYSFUN.

Devuelve la tangente del arco del argumento como un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo de datos incorporado. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

ATAN2

►►—ATAN2—(—*expresión*—,—*expresión*—)———————►►

El esquema es SYSFUN.

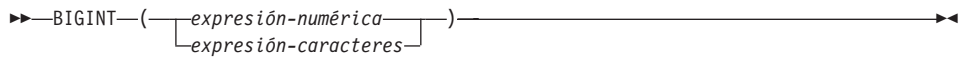
Devuelve la tangente del arco de las coordenadas x e y como un ángulo expresado en radianes. Las coordenadas x e y se especifican por el primer argumento y el segundo respectivamente.

El primer argumento y el segundo pueden ser de cualquier tipo de datos numérico incorporado. Ambos se convierte a un número de coma flotante de precisión doble para que los procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

BIGINT

BIGINT



El esquema es SYSIBM.

La función BIGINT devuelve una representación de entero de 64 bits de un número o serie de caracteres en el formato de una constante de enteros.

expresión-numérica

Una expresión que devuelve un valor de cualquier tipo de datos numérico incorporado.

Si el argumento es una *expresión-numérica*, el resultado es el mismo número que sería si el argumento se asignase a una columna o variable de enteros superiores. Si la parte correspondiente a los enteros del argumento no está dentro del rango de enteros, se produce un error. La parte correspondiente a los decimales del argumento se trunca si está presente.

expresión-caracteres

Una expresión que devuelve un valor de serie de caracteres de longitud no mayor que la longitud máxima de una constante de caracteres. Se eliminan los blancos iniciales y de cola y la serie resultante debe ajustarse a las reglas para la formación de una constante de enteros SQL (SQLSTATE 22018). La serie de caracteres no puede ser una serie larga.

Si el argumento es una *expresión-caracteres*, el resultado es el mismo número que sería si la constante de enteros correspondiente se asignase a una columna o variable de enteros superiores.

El resultado de la función es un entero superior. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplos:

- En la tabla ORDERS_HISTORY, cuente el número de órdenes y devuelva el resultado como un valor de entero superior.

```
SELECT BIGINT (COUNT_BIG(*) )  
FROM ORDERS_HISTORY
```

- Utilizando la tabla EMPLOYEE, seleccione la columna EMPNO en el formato de enteros superiores para procesarla más en la aplicación.

```
SELECT BIGINT(EMPNO) FROM EMPLOYEE
```

BLOB

►► BLOB (—*expresión-serie*—, —*entero*—) ►►

El esquema es SYSIBM.

La función BLOB devuelve una representación BLOB de una serie de cualquier tipo.

expresión-serie

Una *expresión-serie* cuyo valor puede ser una serie de caracteres, una serie gráfica o una serie binaria.

entero

Un valor entero que especifica el atributo de longitud del tipo de datos BLOB resultante. Si no se especifica *entero*, el atributo de longitud del resultado es el mismo que la longitud de la entrada, excepto cuando la entrada es gráfica. En este caso, el atributo de longitud del resultado es el doble de la longitud de la entrada.

El resultado de la función es un BLOB. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

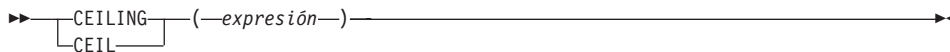
Ejemplos

- Suponiendo una tabla con una columna BLOB denominada TOPOGRAPHIC_MAP y una columna VARCHAR denominada MAP_NAME, localice los mapas que contienen la serie 'Pellow Island' y devuelva una sola serie binaria con el nombre del mapa concatenado delante del mapa real.

```
SELECT BLOB(MAP_NAME || ': ') || TOPOGRAPHIC_MAP
FROM ONTARIO_SERIES_4
WHERE TOPOGRAPIC_MAP LIKE BLOB('%Pellow Island%')
```

CEILING o CEIL

CEILING o CEIL



El esquema es SYSFUN.

Devuelve el valor del entero más pequeño que es mayor o igual que el argumento.

El argumento puede ser de cualquier tipo numérico incorporado. Si el argumento tiene el tipo DECIMAL o REAL, se convierte a un número de coma flotante de precisión doble para que lo procese la función. Si el argumento es del tipo SMALLINT o INTEGER, se devuelve el valor del argumento.

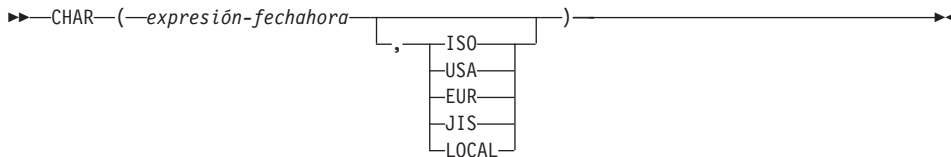
El resultado de la función es:

- SMALLINT si el argumento es SMALLINT
- INTEGER si el argumento es INTEGER
- BIGINT si el argumento es BIGINT
- DOUBLE si el argumento es DECIMAL, REAL o DOUBLE. Los valores decimales con más de 15 dígitos a la izquierda del decimal no devolverán el valor entero deseado debido a que pierden precisión en la conversión a DOUBLE.

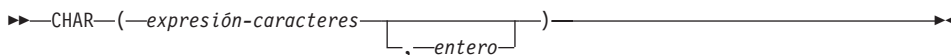
El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

CHAR

De fecha y hora a caracteres :



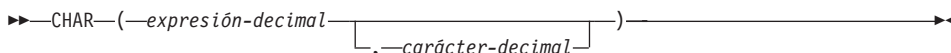
De caracteres a caracteres :



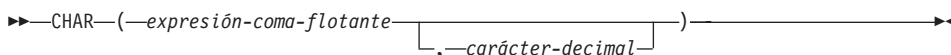
De entero a carácter:



De decimal a carácter:



De coma flotante a caracteres:



El esquema es SYSIBM. Sin embargo, el esquema para CHAR(expresión-coma-flotante) es SYSFUN.

La función CHAR devuelve una representación de serie de caracteres de longitud fija de:

- Un valor de indicación de fecha y hora si el primer argumento es una fecha, una hora o una indicación de la hora
- Una serie de caracteres si el primer argumento es cualquier tipo de serie de caracteres
- Un número entero si el primer argumento es SMALLINT, INTEGER o BIGINT

CHAR

- Un número decimal si el primer argumento es un número decimal
- Un número de coma flotante de precisión doble si el primer argumento es DOUBLE o REAL.

El primer argumento debe ser de un tipo de datos incorporado.

Nota: La expresión CAST también se puede utilizar para devolver una expresión-serie. Para obtener más información, consulte el apartado “Especificaciones CAST” en la página 193.

El resultado de la función es una serie de caracteres de longitud fija. Si el primer argumento puede ser nulo, el resultado puede ser nulo. Si el primer argumento es nulo, el resultado es el valor nulo.

De fecha y hora a caracteres

expresión-fechahora

Una expresión que es de uno de los tres tipos de datos siguientes

fecha El resultado es la representación de serie de caracteres de la fecha en el formato especificado por el segundo argumento. La longitud del resultado es 10. Se produce un error si se especifica el segundo argumento y no es un valor válido (SQLSTATE 42703).

hora El resultado es la representación de serie de caracteres de la hora en el formato especificado por el segundo argumento. La longitud del resultado es 8. Se produce un error si se especifica el segundo argumento y no es un valor válido (SQLSTATE 42703).

indicación de la hora

El segundo argumento no es aplicable y no se debe especificar (SQLSTATE 42815). El resultado es la representación de serie de caracteres de la indicación de la fecha y hora. La longitud del resultado es 26.

La página de códigos de la serie es la página de códigos de la base de datos en el servidor de aplicaciones.

De caracteres a caracteres

expresión-caracteres

Una expresión que devuelve un valor que es de un tipo de datos CHAR, VARCHAR, LONG VARCHAR o CLOB.

entero

el atributo de longitud para la serie de caracteres de longitud fija resultante. El valor debe estar entre 0 y 254.

Si la longitud de la expresión-caracteres es menor que el atributo de longitud del resultado, el resultado se rellena con blancos hasta la longitud del resultado. Si la longitud de la expresión-caracteres es mayor que el atributo de longitud del resultado, se lleva a cabo un truncamiento. Se devuelve un error (SQLSTATE 01004) a menos que todos los caracteres truncados sean blancos y la expresión-caracteres no fuese una serie larga (LONG VARCHAR o CLOB).

De entero a carácter

expresión-entero

Una expresión que devuelve un valor que es de un tipo de datos entero (SMALLINT, INTEGER o BIGINT).

El resultado es la representación de serie de caracteres del argumento en el formato de una constante de enteros de SQL. El resultado consta de n caracteres que son dígitos significativos que representan el valor del argumento con un signo menos que lo precede si el argumento es negativo. Se justifica por la izquierda.

- Si el primer argumento es un entero pequeño:

La longitud del resultado es 6. Si el número de caracteres del resultado es menor que 6, entonces el resultado se rellena por la derecha con blancos hasta la longitud de 6.

- Si el primer argumento es un entero grande:

La longitud del resultado es 11. Si el número de caracteres del resultado es menor que 11, entonces el resultado se rellena por la derecha con blancos hasta la longitud de 11.

- Si el primer argumento es un entero superior:

La longitud del resultado es 20. Si el número de caracteres del resultado es menor que 20, el resultado se rellena por la derecha con blancos hasta la longitud de 20.

La página de códigos de la serie es la página de códigos de la base de datos en el servidor de aplicaciones.

De decimal a carácter

expresión-decimal

Una expresión que devuelve un valor que es de un tipo de datos decimal. Si se desean una precisión y escala diferentes, puede utilizarse primero la función escalar DECIMAL para realizar el cambio.

carácter-decimal

Especifica la constante de caracteres de un solo byte que se utiliza para delimitar los dígitos decimales en la serie de caracteres del resultado. El carácter no puede ser un dígito, el signo más ('+'), el

signo menos ('-') ni un espacio en blanco (SQLSTATE 42815). El valor por omisión es el carácter punto ('.').

El resultado es la representación de serie de caracteres de longitud fija del argumento. El resultado incluye un carácter decimal y p dígitos, donde p es la precisión de la *expresión-decimal* con un signo menos precedente si el argumento es negativo. La longitud del resultado es $2+p$, donde p es la precisión de la *expresión-decimal*. Esto significa que un valor positivo siempre incluirá un blanco de cola.

La página de códigos de la serie es la página de códigos de la base de datos en el servidor de aplicaciones.

De coma flotante a caracteres

expresión-coma-flotante

Una expresión que devuelve un valor que es de un tipo de datos de coma flotante (DOUBLE o REAL).

carácter-decimal

Especifica la constante de caracteres de un solo byte que se utiliza para delimitar los dígitos decimales en la serie de caracteres del resultado. El carácter no puede ser un dígito, el signo más ('+'), el signo menos ('-') ni un espacio en blanco (SQLSTATE 42815). El valor por omisión, utilizando la *signatura SYSFUN.CHAR(expresión-coma-flotante)*, se basa en el entorno nacional del servidor de bases de datos (el valor por omisión puede ser un carácter de punto ('.') o coma (',')).³⁴

El resultado es la representación de serie de caracteres de longitud fija del argumento en la forma de una constante de coma flotante. La longitud del resultado es 24. Si el argumento es negativo, el primer carácter del resultado es un signo menos. De lo contrario, el primer carácter es un dígito. Si el valor del argumento es cero, el resultado es 0E0. De lo contrario, el resultado incluye el número más pequeño de caracteres que puedan representar el valor del argumento de tal modo que la mantisa conste de un solo dígito distinto de cero seguido del *carácter-decimal* y una secuencia de dígitos. Si el número de caracteres del resultado es menor que 24, el resultado se rellena por la derecha con blancos hasta la longitud de 24.

34. En el futuro, se introducirá una *signatura SYSIBM.CHAR(expresión-coma-flotante)* que utilizará por omisión el punto como carácter decimal, independientemente del entorno nacional del servidor de bases de datos. Aunque *SYSFUN.CHAR(expresión-coma-flotante)* continuará estando disponible, la vía de acceso de SQL por omisión haría que la resolución de función utilizara la versión SYSIBM de la función. Si se desea obtener el comportamiento por omisión basado en el entorno de base de datos, se recomienda especificar explícitamente el nombre de esquema SYSFUN al utilizar *SYSFUN.CHAR(expresión-coma-flotante)*.

La página de códigos de la serie es la página de códigos de la base de datos en el servidor de aplicaciones.

Ejemplos:

- Supongamos que la columna PRSTDATE tiene un valor interno equivalente a 1988-12-25.

```
CHAR(PRSTDATE, USA)
```

Da como resultado el valor '12/25/1988'.

- Suponga que la columna STARTING tiene un valor interno equivalente a 17:12:30, la variable del lenguaje principal HOUR_DUR (decimal(6,0)) es una duración en horas con un valor de 050000, (es decir, 5 horas).

```
CHAR(STARTING, USA)
```

Da como resultado el valor '5:12 PM'.

```
CHAR(STARTING + :HOUR_DUR, USA)
```

Da como resultado el valor '10:12 PM'.

- Suponga que la columna RECEIVED (indicación de la hora) tiene un valor interno equivalente a la combinación de las columnas PRSTDATE y STARTING.

```
CHAR(RECEIVED)
```

Da como resultado el valor '1988-12-25-17.12.30.000000'.

- Utilice la función CHAR para que el tipo sea de caracteres de longitud fija y para reducir la longitud de los resultados visualizados a 10 caracteres para la columna LASTNAME (definido como VARCHAR(15)) de la tabla EMPLOYEE.

```
SELECT CHAR(LASTNAME,10) FROM EMPLOYEE
```

Para filas que tengan LASTNAME con una longitud mayor que 10 caracteres (excluyendo los blancos de cola), se devuelve un aviso de que el valor se ha truncado.

- Utilice la función CHAR para devolver los valores para EDLEVEL (definido como smallint) como una serie de caracteres de longitud fija.

```
SELECT CHAR(EDLEVEL) FROM EMPLOYEE
```

Un EDLEVEL de 18 se devolvería como el valor CHAR(6) '18 ' (18 seguido de cuatro blancos).

- Suponga que STAFF tiene una columna SALARY definida como decimal con la precisión 9 y la escala 2. El valor actual es 18357.50 y se debe visualizar con una coma decimal (18357,50).

```
CHAR(SALARY, ',')
```

CHAR

devuelve el valor '00018357,50 '.

- Suponga que la misma columna SALARY que se resta de 20000.25 se ha de visualizar con el carácter decimal por omisión.

```
CHAR(20000.25 - SALARY)
```

devuelve el valor '-0001642.75'.

- Suponga que la variable del lenguaje principal, SEASONS_TICKETS, tiene un tipo de datos entero y un valor 10000.

```
CHAR(DECIMAL(:SEASONS_TICKETS,7,2))
```

Da como resultado el valor de caracteres '10000.00 '.

- Suponga una variable del lenguaje principal, DOUBLE_NUM, que tiene un tipo de datos doble y un valor de -987.654321E-35.

```
CHAR(:DOUBLE_NUM)
```

Da como resultado el valor de caracteres '-9.87654321E-33 '. Puesto que el tipo de datos del resultado es CHAR(24), hay 9 blancos de cola en el resultado.

CHR

►►—CHR—(*—expresión—*)—►►

El esquema es SYSFUN.

Devuelve el carácter que tiene el valor del código ASCII especificado por el argumento.

El argumento puede ser INTEGER o SMALLINT. El valor del argumento debe estar entre 0 y 255; de lo contrario, el valor de retorno es nulo.

El resultado de la función es CHAR(1). El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

CLOB

CLOB

► CLOB (*expresión-serie-caracteres* [, *entero*]) ►

El esquema es SYSIBM.

La función CLOB devuelve una representación CLOB de un tipo de serie de caracteres.

expresión-serie-caracteres

Una *expresión* que devuelve un valor que es una serie de caracteres.

entero

Un valor entero que especifica el atributo de longitud del tipo de datos CLOB resultante. El valor debe estar entre 0 y 2 147 483 647. Si no se especifica *entero*, la longitud del resultado es la misma que la longitud del primer argumento.

El resultado de la función es CLOB. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

COALESCE

►► COALESCE ⁽¹⁾ (—expresión—, —expresión—) ►►

Notas:

1 VALUE es sinónimo de COALESCE.

El esquema es SYSIBM.

COALESCE devuelve el primer argumento que no es nulo.

Los argumentos se evalúan en el orden en que se especifican, y el resultado de la función es el primer argumento que no es nulo. El resultado sólo puede ser nulo si todos los argumentos pueden ser nulos, y el resultado sólo es nulo si todos los argumentos son nulos. El argumento seleccionado se convierte, si es necesario, a los atributos del resultado.

Los argumentos deben ser compatibles. Consulte el apartado “Reglas para los tipos de datos del resultado” en la página 120 para ver qué tipos de datos son compatibles y los atributos del resultado. Puede ser un tipo de datos incorporado o un tipo de datos definido por el usuario.³⁵

Ejemplos:

- Cuando se seleccionan todos los valores de todas las filas de la tabla DEPARTMENT, si falta el director del departamento (MGRNO) (es decir, es nulo), se ha de devolver un valor de 'ABSENT'.

```
SELECT DEPTNO, DEPTNAME, COALESCE(MGRNO, 'ABSENT'), ADMRDEPT
FROM DEPARTMENT
```

- Cuando se selecciona el número de empleado (EMPNO) y el salario (SALARY) de todas las filas de la tabla EMPLOYEE, si falta el salario (es decir, es nulo), se ha de devolver un valor de cero.

```
SELECT EMPNO, COALESCE(SALARY, 0)
FROM EMPLOYEE
```

35. Esta función no puede utilizarse como una función fuente cuando se crea una función definida por el usuario. Como acepta como argumento cualquier tipo de datos compatible, no es necesario crear firmas adicionales para dar soporte a los tipos diferenciados definidos por el usuario.

CONCAT

CONCAT

(1)
▶—CONCAT—(—*expresión1*—,—*expresión2*—)————▶

Notas:

1 || se utiliza como sinónimo de CONCAT.

El esquema es SYSIBM.

Devuelve la concatenación de dos argumentos de serie. Los dos argumentos deben ser de tipos compatibles.

El resultado de la función es una serie. La longitud es la suma de las longitudes de los dos argumentos. Si cualquier argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Vea “Con el operador de concatenación” en la página 177 para obtener más información.

COS

►►—COS—(—*expresión*—)———►►

El esquema es SYSFUN.

Devuelve el coseno del argumento, donde el argumento es un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo numérico incorporado. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

COT

COT

► COT(*—expresión—*) ◄

El esquema es SYSFUN.

Devuelve la cotangente del argumento, donde el argumento es un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo numérico incorporado. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

DATE

►►—DATE—(—*expresión*—)—————►►

El esquema es SYSIBM.

La función DATE devuelve una fecha de un valor.

El argumento debe ser una fecha, indicación de la hora, número positivo menor o igual que 3 652 059, representación de serie de caracteres válida de fecha o indicación de la hora o una serie de caracteres de longitud 7 que no sea CLOB ni LONG VARCHAR.

Si el argumento es una serie de caracteres de longitud 7, debe representar una fecha válida en el formato *aaaannnn*, donde *aaaa* son dígitos que indican un año y *nnn* son dígitos entre 001 y 366, que indican un día de dicho año.

El resultado de la función es una fecha. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Las demás reglas dependen del tipo de datos del argumento:

- Si el argumento es una fecha, una indicación de la hora o una representación de una fecha o indicación de la hora:
 - El resultado es la parte correspondiente a la fecha del valor.
- Si el argumento es un número:
 - El resultado es la fecha de *n*-1 días después de 1 de enero de 0001, donde *n* es la parte integral del número.
- Si el argumento es una serie de caracteres con una longitud de 7:
 - El resultado es la fecha representada por la serie de caracteres.

Ejemplos:

- Suponga que la columna RECEIVED (indicación de la hora) tiene un valor interno equivalente a '1988-12-25-17.12.30.000000'.

DATE(RECEIVED)

El resultado es una representación interna de '1988-12-25'.

- Este ejemplo da como resultado una representación interna de '1988-12-25'.
DATE('1988-12-25')
- Este ejemplo da como resultado una representación interna de '1988-12-25'.
DATE('25.12.1988')
- Este ejemplo da como resultado una representación interna de '0001-02-04'.

DATE

DATE(35)

DAY

►►—DAY—(—*expresión*—)—————►►

El esquema es SYSIBM.

La función DAY devuelve la parte correspondiente al día de un valor.

El argumento debe ser una fecha, una indicación de la hora, una duración de fecha, una duración de indicación de la hora o una representación de serie de caracteres válida de una fecha o indicación de la hora que no sea CLOB ni LONG VARCHAR.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Las demás reglas dependen del tipo de datos del argumento:

- Si el argumento es una fecha, una indicación de la hora o una representación de una fecha o indicación de la hora:
 - El resultado es la parte correspondiente al día del valor, que es un entero entre 1 y 31.
- Si el argumento es una duración de fecha o duración de indicación de la hora:
 - El resultado es la parte correspondiente al día del valor, que es un entero entre -99 y 99. El resultado que no es cero tiene el mismo signo que el argumento.

Ejemplos:

- Utilizando la tabla PROJECT, establezca la variable del lenguaje principal END_DAY (smallint) en el día en que está planificado que el proyecto WELD LINE PLANNING (PROJNAME) finalice (PRENDATE).

```
SELECT DAY(PRENDATE)
INTO :END_DAY
FROM PROJECT
WHERE PROJNAME = 'WELD LINE PLANNING'
```

Da como resultado que END_DAY se establece en 15 cuando se utiliza la tabla de ejemplo.

- Supongamos que la columna DATE1 (fecha) tiene un valor interno equivalente a 2000-03-15 y la columna DATE2 (fecha) tiene un valor interno equivalente a 1999-12-31.

```
DAY (DATE1 - DATE2)
```

DAY

Da como resultado el valor 15.

DAYNAME

►►—DAYNAME—(*—expresión—*)—►►

El esquema es SYSFUN.

Devuelve una serie de caracteres en mayúsculas y minúsculas mezcladas que contienen el nombre del día (p. ej., Viernes) para la parte del día del argumento basándose en el entorno nacional del momento en que se ha iniciado la base de datos.

El argumento debe ser una fecha, una indicación de la hora o una representación de serie de caracteres válida de una fecha o indicación de la hora que no sea CLOB ni LONG VARCHAR.

El resultado de la función es VARCHAR(100). El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

DAYOFWEEK

DAYOFWEEK

►—DAYOFWEEK—(*—expresión—*)—◄

El esquema es SYSFUN.

Devuelve el día de la semana del argumento como un valor entero en el rango de 1 a 7, donde 1 representa el Domingo.

El argumento debe ser una fecha, una indicación de la hora o una representación de serie de caracteres válida de una fecha o indicación de la hora que no sea CLOB ni LONG VARCHAR.

El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

DAYOFWEEK_ISO

►►—DAYOFWEEK_ISO—(*—expresión—*)——————►►

El esquema es SYSFUN.

Devuelve el día de la semana del argumento, en forma de valor entero comprendido dentro del rango 1-7, donde 1 representa el lunes.

El argumento debe ser una fecha, una indicación de la hora o una representación de serie de caracteres válida de una fecha o indicación de la hora que no sea CLOB ni LONG VARCHAR.

El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

DAYOFYEAR

DAYOFYEAR

►—DAYOFYEAR—(*—expresión—*)—◄

El esquema es SYSFUN.

Devuelve el día del año del argumento como un valor entero en el rango de 1 a 366.

El argumento debe ser una fecha, una indicación de la hora o una representación de serie de caracteres válida de una fecha o indicación de la hora que no sea CLOB ni LONG VARCHAR.

El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

DAYS

►►—DAYS—(—expresión—)—————►►

El esquema es SYSIBM.

La función DAYS devuelve una representación de entero de una fecha.

El argumento debe ser una fecha, una indicación de la hora o una representación de serie de caracteres válida de una fecha o indicación de la hora que no sea CLOB ni LONG VARCHAR.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

El resultado es 1 más que el número de días desde el 1 de enero de 0001 hasta *D*, donde *D* es la fecha que podría darse si se aplicase la función DATE al argumento.

Ejemplos:

- Utilizando la tabla PROJECT, establezca la variable del lenguaje principal EDUCATION_DAYS (int) en el número de días transcurridos (PRENDATE - PRSTDATE) estimados para el proyecto (PROJNO) 'IF2000'.

```
SELECT DAYS(PRENDATE) - DAYS(PRSTDATE)
INTO :EDUCATION_DAYS
FROM PROJECT
WHERE PROJNO = 'IF2000'
```

Da como resultado que EDUCATION_DAYS se establece en 396 cuando se utiliza la tabla de ejemplo.

- Utilizando la tabla PROJECT, establezca la variable del lenguaje principal TOTAL_DAYS (int) en la suma de los días transcurridos (PRENDATE - PRSTDATE) estimados para todos los proyectos del departamento (DEPTNO) 'E21'.

```
SELECT SUM(DAYS(PRENDATE) - DAYS(PRSTDATE))
INTO :TOTAL_DAYS
FROM PROJECT
WHERE DEPTNO = 'E21'
```

Da como resultado que TOTAL_DAYS se establece en 1584 cuando se utiliza la tabla de ejemplo.

DBCLOB

DBCLOB

►► DBCLOB (—*expresión-gráfica* [—*entero*—]) ►►

El esquema es SYSIBM.

La función DBCLOB devuelve una representación DBCLOB de un tipo de serie gráfica.

expresión-gráfica

Una *expresión* que devuelve un valor que es una serie gráfica.

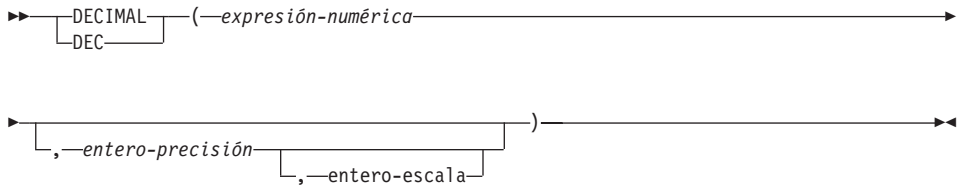
entero

Un valor entero que especifica el atributo de longitud del tipo de datos DBCLOB resultante. El valor debe estar entre 0 y 1 073 741 823. Si no se especifica *entero*, la longitud del resultado es la misma que la longitud del primer argumento.

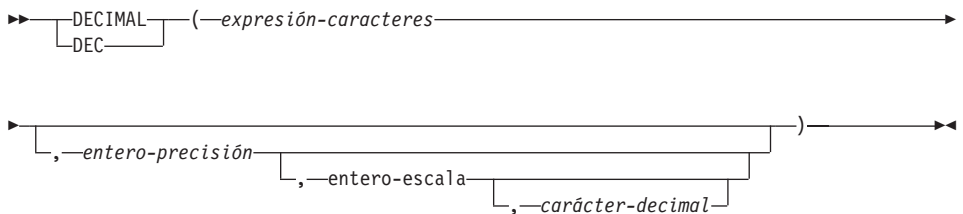
El resultado de la función es DBCLOB. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

DECIMAL

De numérico a decimal :



De carácter a decimal:



El esquema es SYSIBM.

La función DECIMAL devuelve una representación decimal de

- Un número
- Una representación de serie de caracteres de un número decimal
- Una representación de serie de caracteres de un número entero.

El resultado de la función es un número decimal con la precisión de p y la escala de s , donde p y s son el segundo y el tercer argumento. Si el primer argumento puede ser nulo, el resultado puede ser nulo; si el primer argumento es nulo, el resultado es el valor nulo.

De numérico a decimal

expresión-numérica

Una expresión que devuelve un valor de cualquier tipo de datos numérico.

entero-precisión

Una constante de enteros con un valor en el rango de 1 a 31.

El valor por omisión para el *entero-precisión* depende del tipo de datos de la *expresión-numérica*:

- 15 para coma flotante y decimal

DECIMAL

- 19 para entero superior
- 11 para entero grande
- 5 para entero pequeño.

entero-escala

Una constante de enteros en el rango de 0 al valor de *entero-precisión*. El valor por omisión es cero.

El resultado es el mismo número que sería si se asignase el primer argumento a una columna o variable decimal con una precisión de p y una escala de s , donde p y s son el segundo y el tercer argumento. Se produce un error si el número de dígitos decimales significativos necesarios para representar la parte correspondiente a los enteros es mayor que $p-s$.

De carácter a decimal

expresión-caracteres

Una *expresión* que devuelve un valor que es una serie de caracteres con una longitud no mayor que la longitud máxima de una constante de caracteres (4.000 bytes). No puede tener un tipo de datos CLOB ni LONG VARCHAR. Los blancos iniciales o de cola se eliminan de la serie. La subserie resultante debe ajustarse a las reglas para formar una constante decimal o de entero SQL (SQLSTATE 22018).

La *expresión-caracteres* se convierte a la página de códigos de la base de datos si es necesario que coincida con la página de códigos de la constante *carácter-decimal*.

entero-precisión

Una constante de enteros con un valor en el rango de 1 a 31 que especifica la precisión del resultado. Si no se especifica, el valor por omisión es 15.

entero-escala

Una constante de enteros con un valor en el rango de 0 al *entero-precisión* que especifica la escala del resultado. Si no se especifica, el valor por omisión es 0.

carácter-decimal

Especifica la constante de caracteres de un solo byte utilizada para delimitar los dígitos decimales en *expresión-caracteres* de la parte correspondiente a los enteros del número. El carácter no puede ser un dígito, el signo más ('+'), el signo menos ('-') ni un espacio en blanco y puede aparecer como máximo una vez en *expresión-caracteres* (SQLSTATE 42815).

El resultado es un número decimal con la precisión p y la escala s donde p y s son el segundo y el tercer argumento. Los dígitos se truncan por el final del número decimal si el número de dígitos a la derecha del carácter decimal es mayor que la escala s . Se produce un error si el número de dígitos significativos a la izquierda del carácter decimal (la parte correspondiente a los enteros del número) de *expresión-caracteres* es mayor que $p-s$ (SQLSTATE 22003). El carácter decimal por omisión no es válido en la subserie si se especifica un valor diferente para el argumento *carácter-decimal* (SQLSTATE 22018).

Ejemplos:

- Utilice la función DECIMAL para forzar a que se devuelva un tipo de datos DECIMAL (con una precisión de 5 y una escala de 2) en una lista-selección para la columna EDLEVEL (tipo de datos = SMALLINT) en la tabla EMPLOYEE. La columna EMPNO debe aparecer también en la lista de selección.

```
SELECT EMPNO, DECIMAL(EDLEVEL,5,2)
      FROM EMPLOYEE
```

- Suponga que la variable del lenguaje principal PERIOD es de tipo INTEGER. En este caso, para utilizar su valor como duración de fecha debe convertirse a decimal(8,0).

```
SELECT PRSTDATE + DECIMAL(:PERIOD,8)
      FROM PROJECT
```

- Suponga que las actualizaciones en la columna SALARY se entran mediante una ventana como una serie de caracteres que utiliza la coma como carácter decimal (por ejemplo, el usuario entra 21400,50). Cuando se ha validado por la aplicación, se asigna a la variable del lenguaje principal newsalary definida como CHAR(10).

```
UPDATE STAFF
  SET SALARY = DECIMAL(:newsalary, 9, 2, ',',')
  WHERE ID = :empid;
```

El valor de newsalary se convierte en 21400.50.

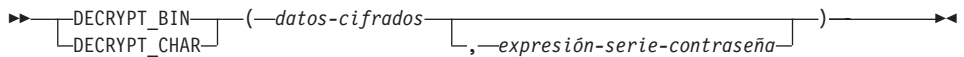
- Añada el carácter decimal por omisión (.) al valor.

```
DECIMAL('21400,50', 9, 2, '.',')
```

Falla porque se especifica un punto (.) como el carácter decimal pero aparece una coma (,) en el primer argumento como delimitador.

DECRYPT_BIN y DECRYPT_CHAR

DECRYPT_BIN y DECRYPT_CHAR



El esquema es SYSIBM.

Las funciones DECRYPT_BIN y DECRYPT_CHAR devuelven un valor que es el resultado del descifrado de *datos-cifrados*. La contraseña utilizada para el descifrado es el valor de *expresión-serie-contraseña* o el valor de ENCRYPTION PASSWORD asignado por la sentencia SET ENCRYPTION PASSWORD. Las funciones DECRYPT_BIN y DECRYPT_CHAR sólo pueden descifrar valores que se han cifrado utilizando la función ENCRYPT (SQLSTATE 428FE).

datos-cifrados

Una expresión que devuelve un valor CHAR FOR BIT DATA o VARCHAR FOR BIT DATA como una serie de datos cifrada completa. La serie de datos se tiene que haber cifrado utilizando la función ENCRYPT.

expresión-serie-contraseña

Una expresión que devuelve un valor CHAR o VARCHAR con un mínimo de 6 bytes y no más de 127 bytes (SQLSTATE 428FC). Esta expresión debe ser la misma contraseña utilizada para cifrar los datos o, de lo contrario, el descifrado producirá un error (SQLSTATE 428FD). Si el valor del argumento de contraseña es nulo o no se proporciona, los datos se cifrarán utilizando el valor de ENCRYPTION PASSWORD, que tiene que haberse establecido para la sesión.

El resultado de la función DECRYPT_BIN es VARCHAR FOR BIT DATA. El resultado de la función DECRYPT_CHAR es VARCHAR. Si los *datos-cifrados* incluían una indicación, la función no devuelve la indicación. El atributo de longitud del resultado es la longitud del tipo de datos de los *datos-cifrados* menos 8 bytes. La longitud real del valor devuelto por la función coincidirá con la longitud de la serie original que se ha cifrado. Si *datos-cifrados* incluye bytes más allá de la serie cifrada, la función no devolverá estos bytes.

Si el primer argumento puede ser nulo, el resultado puede ser nulo. Si el primer argumento es nulo, el resultado es el valor nulo.

Si los datos se descifran en un sistema diferente que utiliza una página de códigos diferente de la página de códigos en la que se han cifrado los datos, puede que se produzca una expansión al convertir el valor descifrado a la página de códigos de la base de datos. En dichas situaciones, el valor *datos-cifrados* debe calcularse en una serie VARCHAR con un número mayor de bytes.

Consulte el apartado “ENCRYPT” en la página 331 y el apartado “GETHINT” en la página 338 para obtener información adicional sobre cómo utilizar esta función.

Ejemplos:

Ejemplo 1: Este ejemplo utiliza el valor de ENCRYPTION PASSWORD para retener la contraseña de cifrado.

```
SET ENCRYPTION PASSWORD = 'Ben123';
INSERT INTO EMP(SSN) VALUES ENCRYPT('289-46-8832');
SELECT DECRYPT_CHAR(SSN)
FROM SSN;
```

Esto devuelve el valor '289-46-8832'.

Ejemplo 2: Este ejemplo pasa explícitamente la contraseña de cifrado.

```
INSERT INTO EMP (SSN) VALUES ENCRYPT('289-46-8832', 'Ben123', '');
SELECT DECRYPT(SSN, 'Ben123')
FROM SSN;
```

Este ejemplo devuelve el valor '289-46-8832'.

DEGREES

DEGREES

►►—DEGREES—(*—expresión—*)——————►◄

El esquema es SYSFUN.

Devuelve el número de grados convertidos del argumento expresado en radianes.

El argumento puede ser de cualquier tipo numérico incorporado. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

DEREF

►►—DEREF—(—*expresión*—)—————►►

El esquema es SYSIBM.

La función Deref devuelve una instancia del tipo de destino del argumento.

El argumento puede ser cualquier valor con un tipo de datos de referencia que tenga un ámbito definido (SQLSTATE 428DT).

El tipo de datos estático del resultado es el tipo de destino del argumento. El tipo de datos dinámico del resultado es un subtipo del tipo de destino del argumento. El resultado puede ser nulo. El resultado es un valor nulo si *expresión* es un valor nulo o si *expresión* es una referencia que no tiene un OID correspondiente en la tabla de destino.

El resultado es una instancia del subtipo del tipo de destino de la referencia. El resultado se determina buscando la fila de la tabla de destino o vista de destino de la referencia que tenga un identificador de objeto que se corresponda con el valor de la referencia. El tipo de esta fila determina el tipo dinámico del resultado. Puesto que el tipo del resultado puede estar basado en una fila de una subtabla o subvista de la tabla de destino o vista de destino, el ID de autorización de la sentencia debe tener un privilegio SELECT sobre la tabla de destino y todas sus subtablas o sobre la vista de destino y todas sus subvistas (SQLSTATE 42501).

Ejemplos:

Suponga que EMPLOYEE es una tabla de tipo EMP, y que su columna de identificador de objeto se llama EMPID. En este caso, la consulta siguiente devuelve un objeto de tipo EMP (o uno de sus subtipos) para cada fila de la tabla EMPLOYEE (y de sus subtablas). Para ejecutar esta consulta es necesario tener privilegio SELECT sobre EMPLOYEE y todas sus subtablas.

```
SELECT Deref(EMPID) FROM EMPLOYEE
```

Para ver más ejemplos, consulte "TYPE_NAME" en la página 446.

DIFFERENCE

DIFFERENCE

►—DIFFERENCE—(—*expresión*—,—*expresión*—)—►

El esquema es SYSFUN.

Devuelve un valor de 0 a 4 que representa la diferencia entre los sonidos de dos series basándose en la aplicación de la función SOUNDEX en las series. El valor 4 es la mejor coincidencia de sonido posible.

Los argumentos pueden ser series de caracteres que sean CHAR o VARCHAR de un máximo de 4.000 bytes.

El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplo:

```
VALUES (DIFFERENCE('CONSTRAINT', 'CONSTANT'), SOUNDEX('CONSTRAINT'),  
        SOUNDEX('CONSTANT')),  
(DIFFERENCE('CONSTRAINT', 'CONTRITE'), SOUNDEX('CONSTRAINT'),  
        SOUNDEX('CONTRITE'))
```

Este ejemplo devuelve lo siguiente.

1	2	3
-----	-----	-----
	4	C523 C523
	2	C523 C536

En la primera fila, las palabras tienen el mismo resultado de SOUNDEX, mientras que en la segunda fila las palabras sólo tienen algún parecido.

DIGITS

►►—DIGITS—(—*expresión*—)—————►►

El esquema es SYSIBM.

La función DIGITS devuelve una representación de serie de caracteres de un número.

El argumento debe ser una expresión que devuelva un valor con el tipo SMALLINT, INTEGER, BIGINT o DECIMAL.

Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

El resultado de la función es una serie de caracteres de longitud fija que representa el valor absoluto del argumento sin tener en cuenta su escala. El resultado no incluye el signo ni el carácter decimal. En su lugar, consta exclusivamente de dígitos, incluyendo, si es necesario, ceros iniciales para rellenar la serie. La longitud de la serie es:

- 5 si el argumento es un entero pequeño
- 10 si el argumento es un entero grande
- 19 si el argumento es un entero superior
- p si el argumento es un número decimal con una precisión de p .

Ejemplos:

- Suponga que una tabla llamada TABLEX contiene una columna INTEGER llamada INTCOL que contiene números de 10 dígitos. Liste las cuatro combinaciones de dígitos de los cuatro primeros dígitos contenidos en la columna INTCOL.

```
SELECT DISTINCT SUBSTR(DIGITS(INTCOL),1,4)
FROM TABLEX
```

- Suponga que la columna COLUMNX tiene el tipo de datos DECIMAL(6,2) y que uno de sus valores es -6.28. Entonces, para este valor:

```
DIGITS(COLUMNX)
```

devuelve el valor '000628'.

El resultado es una serie de longitud seis (la precisión de la columna) con ceros iniciales que rellenan la serie hasta esta longitud. No aparecen ni el signo ni la coma decimal en el resultado.

DLCOMMENT

DLCOMMENT

►►—DLCOMMENT—(—*expresión-datalink*—)——►►

El esquema es SYSIBM.

La función DLCOMMENT devuelve el valor del comentario, si existe, de un valor DATALINK.

El argumento debe ser una expresión que dé como resultado un valor con el tipo de datos de DATALINK.

El resultado de la función es VARCHAR(254). Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplo:

- Prepare una sentencia para seleccionar la fecha, la descripción y el comentario del enlace de la columna ARTICLES de la tabla HOCKEY_GOALS. Las filas a seleccionar son las correspondientes a los goles marcados por cualquiera de los dos hermanos Richard (Maurice o Henri).

```
stmtvar = "SELECT DATE_OF_GOAL, DESCRIPTION, DLCOMMENT(ARTICLES)
          FROM HOCKEY_GOALS
          WHERE BY_PLAYER = 'Maurice Richard'
          OR BY_PLAYER = 'Henri Richard' ";
EXEC SQL PREPARE HOCKEY_STMT FROM :stmtvar;
```

- Dado un valor DATALINK que se había insertado en la columna COLA de una fila de la tabla TBLA mediante la función escalar:

```
DLVALUE('http://d1fs.almaden.ibm.com/x/y/a.b','URL','Un comentario')
```

la siguiente función que realiza una operación con este valor:

```
DLCOMMENT(COLA)
```

devolverá el valor:

Un comentario

DLLINKTYPE

►►—DLLINKTYPE—(—*expresión-datalink*—)—————►►

El esquema es SYSIBM.

La función DLLINKTYPE devuelve el valor de tipo enlace de un valor DATALINK.

El argumento debe ser una expresión que dé como resultado un valor con el tipo de datos DATALINK.

El resultado de la función es VARCHAR(4). Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplo:

- Dado un valor DATALINK que se había insertado en la columna COLA de una fila de la tabla TBLA mediante la función escalar:

```
DVALUE('http://d1fs.almaden.ibm.com/x/y/a.b', 'URL', 'un comentario')
```

la siguiente función que realiza una operación con este valor:

```
DLLINKTYPE(COLA)
```

devolverá el valor:

```
URL
```

DLURLCOMPLETE

DLURLCOMPLETE

►►—DLURLCOMPLETE—(—*expresión-datalink*—)—————►

El esquema es SYSIBM.

La función DLURLCOMPLETE devuelve el atributo de ubicación de datos a partir de un valor DATALINK, con un tipo de enlace URL. Cuando corresponde, el valor incluye un símbolo de accesos de archivo.

El argumento debe ser una expresión que dé como resultado un valor con el tipo de datos DATALINK.

El resultado de la función es VARCHAR(254). Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Si el valor DATALINK sólo incluye el comentario, el resultado devuelto es una serie de longitud cero.

Ejemplo:

- Dado un valor DATALINK que se había insertado en la columna COLA de una fila de la tabla TBLA mediante la función escalar:

```
DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','un comentario')
```

la siguiente función que realiza una operación con este valor:

```
DLURLCOMPLETE(COLA)
```

devolverá el valor:

```
HTTP://DLFS.ALMA DEN.IBM.COM/x/y/*****;a.b
```

(donde ***** representa el símbolo de accesos)

DLURLPATH

►►—DLURLPATH—(—*expresión-datalink*—)—————►►

El esquema es SYSIBM.

La función DLURLPATH devuelve la vía de acceso y el nombre de archivo necesarios para acceder a un archivo de un servidor determinado desde un valor DATALINK con un tipoenlace de URL. Cuando corresponde, el valor incluye un símbolo de accesos de archivo.

El argumento debe ser una expresión que dé como resultado un valor con el tipo de datos DATALINK.

El resultado de la función es VARCHAR(254). Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Si el valor DATALINK sólo incluye el comentario, el resultado devuelto es una serie de longitud cero.

Ejemplo:

- Dado un valor DATALINK que se había insertado en la columna COLA de una fila de la tabla TBLA mediante la función escalar:

```
DLVALUE('http://d1fs.almaden.ibm.com/x/y/a.b','URL','un comentario')
```

la siguiente función que realiza una operación con este valor:

```
DLURLPATH(COLA)
```

devolverá el valor:

```
/x/y/*****;a.b
```

(donde ***** representa el símbolo de accesos)

DLURLPATHONLY

DLURLPATHONLY

►—DLURLPATHONLY—(*—expresión-datalink—*)—►

El esquema es SYSIBM.

La función DLURLPATHONLY devuelve la vía de acceso y el nombre de archivo necesarios para acceder a un archivo de un servidor determinado desde un valor DATALINK con un tipo de enlace de URL. El valor devuelto NUNCA incluye un símbolo de accesos de archivo.

El argumento debe ser una expresión que dé como resultado un valor con el tipo de datos DATALINK.

El resultado de la función es VARCHAR(254). Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Si el valor DATALINK sólo incluye el comentario, el resultado devuelto es una serie de longitud cero.

Ejemplo:

- Dado un valor DATALINK que se había insertado en la columna COLA de una fila de la tabla TBLA mediante la función escalar:

```
DLVALUE('http://d1fs.almaden.ibm.com/x/y/a.b','URL','un comentario')
```

la siguiente función que realiza una operación con este valor:

```
DLURLPATHONLY(COLA)
```

devolverá el valor:

```
/x/y/a.b
```

DLURLSCHEME

►►—DLURLSCHEME—(—*expresión-datalink*—)—————►►

El esquema es SYSIBM.

La función DLURLSCHEME devuelve el esquema de un valor DATALINK con un tipo enlace de URL. El valor siempre estará en mayúsculas.

El argumento debe ser una expresión que dé como resultado un valor con el tipo de datos DATALINK.

El resultado de la función es VARCHAR(20). Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Si el valor DATALINK sólo incluye el comentario, el resultado devuelto es una serie de longitud cero.

Ejemplo:

- Dado un valor DATALINK que se había insertado en la columna COLA de una fila de la tabla TBLA mediante la función `ESCALAR`:

```
DLVALUE('http://dfs.almaden.ibm.com/x/y/a.b','URL','un comentario')
```

la siguiente función que realiza una operación con este valor:

```
DLURLSCHEME(COLA)
```

devolverá el valor:

```
HTTP
```

DLURLSERVER

DLURLSERVER

►—DLURLSERVER—(*—expresión-datalink—*)—◄

El esquema es SYSIBM.

La función DLURLSERVER devuelve el servidor de archivos de un valor DATALINK con un tipo enlace de URL. El valor siempre estará en mayúsculas.

El argumento debe ser una expresión que dé como resultado un valor con el tipo de datos DATALINK.

El resultado de la función es VARCHAR(254). Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Si el valor DATALINK sólo incluye el comentario, el resultado devuelto es una serie de longitud cero.

Ejemplo:

- Dado un valor DATALINK que se había insertado en la columna COLA de una fila de la tabla TBLA mediante la función escalar:

```
DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','un comentario')
```

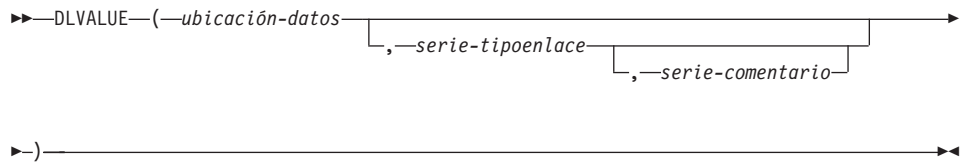
la siguiente función que realiza una operación con este valor:

```
DLURLSERVER(COLA)
```

devolverá el valor:

```
DLFS.ALMADEN.IBM.COM
```

DLVALUE



El esquema es SYSIBM.

La función DLVALUE devuelve un valor DATALINK. Cuando la función está al lado derecho de una cláusula SET dentro de una sentencia UPDATE o está en una cláusula VALUES dentro de una sentencia INSERT, normalmente también crea un enlace con un archivo. No obstante, si sólo se especifica un comentario (en cuyo caso la serie ubicación-datos tiene longitud-cero), el valor DATALINK se crea con atributos de enlace vacíos, por lo que no hay enlace de archivo.

ubicación-datos

Si el tipo de enlace es URL, ésta es una expresión que proporciona una serie de caracteres de longitud variable que contiene un valor de URL completo.

serie-tipoenlace

Una expresión VARCHAR opcional que especifica el tipo de enlace del valor DATALINK. El único valor válido es 'URL' (SQLSTATE 428D1).

serie-comentario

Un valor VARCHAR(254) opcional que proporciona comentarios o información de tipo adicional sobre la ubicación.

El resultado de la función es un valor DATALINK. Si cualquier argumento de la función DLVALUE puede ser nulo, el resultado puede ser nulo; si *ubicación-datos* es nula, el resultado es el valor nulo.

Cuando defina un valor DATALINK mediante esta función, tenga en cuenta la longitud máxima del destino del valor. Por ejemplo, si se define una columna como DATALINK(200), la longitud máxima de *ubicación-datos* más *comentario* suma 200 bytes.

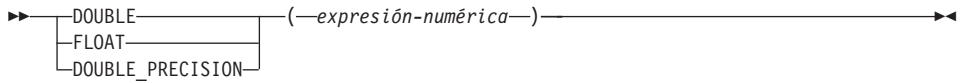
Ejemplo:

- Inserte una fila en la tabla. Los valores de URL para los dos primeros enlaces se incluyen en las variables denominadas `url_article` y `url_snapshot`. La variable denominada `url_snapshot_comment` contiene un comentario que acompaña el enlace de snapshot. Todavía no existe el enlace de movie, sólo un comentario en la variable denominada `url_movie_comment`.

DLVALUE

```
EXEC SQL INSERT INTO HOCKEY_GOALS
VALUES('Maurice Richard',
        'Montreal Canadien',
        '?',
        'Boston Bruins',
        '1952-04-24',
        'Winning goal in game 7 of Stanley Cup final',
        DLVALUE(:url_article),
        DLVALUE(:url_snapshot, 'URL', :url_snapshot_comment),
        DLVALUE('', 'URL', :url_movie_comment) );
```

DOUBLE

De numérico a doble :**De serie de caracteres a doble :**

El esquema es SYSIBM. Sin embargo, el esquema de `DOUBLE(expresión-serie)` es SYSFUN.

La función `DOUBLE` devuelve un número de coma flotante correspondiente a:

- un número si el argumento es una expresión numérica
- una representación de serie de caracteres de un número si el argumento es una expresión de serie.

De numérico a doble*expresión-numérica*

El argumento es una expresión que devuelve un valor de cualquier tipo de datos numérico incorporado.

El resultado de la función es un número de coma flotante de precisión doble. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

El resultado es el mismo número que sería si el argumento se hubiese asignado a una columna o variable de coma flotante de precisión doble.

De serie de caracteres a doble*expresión-serie*

El argumento puede ser de tipo `CHAR` o `VARCHAR` en el formato de una constante numérica. Se ignoran los blancos iniciales y de cola.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

DOUBLE

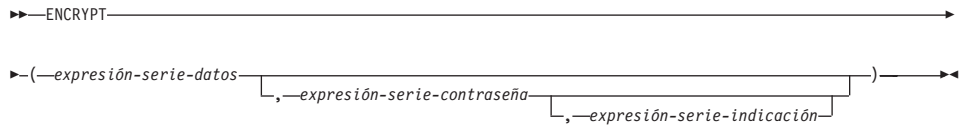
El resultado es el mismo número que sería si la serie se considerase una constante y se asignase a una columna o variable de coma flotante de precisión doble.

Ejemplo:

Utilizando la tabla EMPLOYEE, busque la proporción de salario y comisiones para los empleados cuya comisión no sea cero. Las columnas implicadas (SALARY y COMM) tienen tipos de datos DECIMAL. Para eliminar la posibilidad de resultados fuera de rango, se aplica DOUBLE a SALARY para que la división se lleve a cabo en coma flotante:

```
SELECT EMPNO, DOUBLE(SALARY)/COMM
      FROM EMPLOYEE
 WHERE COMM > 0
```


ENCRYPT



El esquema es SYSIBM.

La función ENCRYPT devuelve un valor que es el resultado del cifrado de *expresión-serie-datos*. La contraseña utilizada para el cifrado es el valor de *expresión-serie-contraseña* o el valor de ENCRYPTION PASSWORD (asignado utilizando la sentencia SET ENCRYPTION PASSWORD).

expresión-serie-datos

Una expresión que devuelve un valor CHAR o VARCHAR que se debe cifrar. El atributo de longitud para el tipo de datos de *expresión-serie-datos* está limitado a 32663 sin ningún argumento *expresión-serie-indicación* y a 32631 cuando se especifica el argumento *expresión-serie-indicación* (SQLSTATE 42815).

expresión-serie-contraseña

Una expresión que devuelve un valor CHAR o VARCHAR con un mínimo de 6 bytes y no más de 127 bytes (SQLSTATE 428FC). El valor representa la contraseña utilizada para cifrar la *expresión-serie-datos*. Si el valor del argumento de contraseña es nulo o no se proporciona, los datos se cifrarán utilizando el valor de ENCRYPTION PASSWORD, que tiene que haberse establecido para la sesión (SQLSTATE 51039).

expresión-serie-indicación

Una expresión que devuelve un valor CHAR o VARCHAR de un máximo de 32 bytes que ayudará a los propietarios de datos a recordar las contraseñas (por ejemplo, 'Océano' como indicación para recordar 'Pacífico'). Si se proporciona un valor de indicación, la indicación se incorpora en el resultado y puede recuperarse utilizando la función GETHINT. Si este argumento es nulo o no se proporciona, no se incorporará ninguna indicación en el resultado.

El tipo de datos de resultado de la función es VARCHAR FOR BIT DATA.

El atributo de longitud del resultado es:

- Cuando se especifica el parámetro de indicación opcional, el atributo de longitud de los datos no cifrados + 8 bytes + el número de bytes hasta el siguiente límite de 8 bytes + 32 bytes para la longitud de la indicación.
- Sin parámetro de indicación, el atributo de longitud de los datos no cifrados + 8 bytes + el número de bytes hasta el siguiente límite de 8 bytes.

ENCRYPT

Si el primer argumento puede ser nulo, el resultado puede ser nulo; si el primer argumento es nulo, el resultado es el valor nulo.

Tenga en cuenta que el resultado cifrado tiene una longitud mayor que la del valor *expresión-serie-datos*. Por consiguiente, al asignar valores cifrados, asegúrese de que el destino se declara con un tamaño suficiente para contener el valor cifrado entero.

Notas:

- **Algoritmo de cifrado:** El algoritmo de cifrado interno utilizado es la cifra de bloque RC2 con relleno, la clave secreta de 128 bits se deriva de la contraseña utilizando una conversión de mensaje MD2.
- **Contraseñas y datos de cifrado** La gestión de contraseñas es responsabilidad del usuario. Una vez que se han cifrado los datos, sólo se puede utilizar para descifrarlos la contraseña utilizada para cifrarlos (SQLSTATE 428FD). Tenga cuidado al utilizar las variables CHAR para establecer valores de contraseña porque pueden estar rellenas con espacios en blanco. El resultado cifrado puede contener el terminador nulo y otros caracteres no imprimibles.
- **Definición de columna de tabla:** Cuando defina columnas y tipos para que contengan datos cifrados, calcule siempre el atributo de longitud del modo siguiente.

Para datos cifrados sin indicación:

Longitud máxima de los datos no cifrados + 8 bytes + el número de bytes hasta el siguiente límite de 8 bytes = longitud de columna de datos cifrados.

Para datos cifrados con indicación incorporada:

Longitud máxima de los datos no cifrados + 8 bytes + el número de bytes hasta el siguiente límite de 8 bytes + 32 bytes para la longitud de la indicación = longitud de columna de datos cifrados.

Cualquier asignación o cálculo a una longitud inferior a la longitud de datos sugerida puede producir un descifrado anómalo en el futuro y hacer que se pierdan datos. Los espacios en blanco son valores de datos cifrados válidos que se pueden truncar al almacenarse en una columna que es demasiado pequeña.

Ejemplos de cálculos de longitud de columna:

Longitud máxima de datos no cifrados	6 bytes
8 bytes	8 bytes
Número de bytes hasta el siguiente límite de 8 bytes	2 bytes

Longitud de columna de datos cifrados	16 bytes
Longitud máxima de datos no cifrados	32 bytes
8 bytes	8 bytes

Número de bytes hasta el siguiente límite de	8 bytes
8 bytes	-----
Longitud de columna de datos cifrados	48 bytes

- **Administración de datos cifrados:** Los datos cifrados sólo se pueden descifrar en servidores que soporten las funciones de descifrado que corresponden a la función ENCRYPT. Por lo tanto, el duplicado de columnas con datos cifrados sólo se debe realizar en servidores que soporten la función DECRYPT_BIN o DECRYPT_CHAR.

Consulte también el apartado “DECRYPT_BIN y DECRYPT_CHAR” en la página 314 y el apartado “GETHINT” en la página 338 para obtener información adicional sobre la utilización de esta función.

Ejemplos:

Ejemplo 1: Este ejemplo utiliza el valor de ENCRYPTION PASSWORD para retener la contraseña de cifrado.

```
SET ENCRYPTION PASSWORD = 'Ben123';
INSERT INTO EMP(SSN) VALUES ENCRYPT('289-46-8832');
```

Ejemplo 2: Este ejemplo pasa explícitamente la contraseña de cifrado.

```
INSERT INTO EMP(SSN) VALUES ENCRYPT('289-46-8832', 'Ben123', '');
```

Ejemplo 3: La indicación ‘Océano’ se almacena para ayudar al usuario a recordar la contraseña de cifrado de ‘Pacífico’.

```
INSERT INTO EMP(SSN) VALUES ENCRYPT('289-46-8832', 'Pacífico', 'Océano');
```

EVENT_MON_STATE

EVENT_MON_STATE

►—EVENT_MON_STATE—(*—expresión-serie—*)—►

El esquema es SYSIBM.

La función EVENT_MON_STATE devuelve el estado actual de un supervisor de sucesos.

El argumento es una expresión de serie con un tipo resultante de CHAR o VARCHAR y un valor que es el nombre de un supervisor de sucesos. Si el supervisor de sucesos nombrado no existe en la tabla del catálogo SYSCAT.EVENTMONITORS, se devolverá SQLSTATE 42704.

El resultado es un entero con uno de los valores siguientes:

- - 0 El supervisor de sucesos está inactivo.
 - 1 El supervisor de sucesos está activo.

Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplo:

- El siguiente ejemplo selecciona todos los supervisores de sucesos definidos e indica si cada uno está activo o inactivo:

```
SELECT EVMONNAME,  
       CASE  
WHEN EVENT_MON_STATE(EVMONNAME) = 0 THEN 'Inactivo'  
     WHEN EVENT_MON_STATE(EVMONNAME) = 1 THEN 'Active'  
     END  
FROM SYSCAT.EVENTMONITORS
```

EXP

►►—EXP—(*—expresión—*)—►►

El esquema es SYSFUN.

Devuelve la función exponencial del argumento.

El argumento puede ser de cualquier tipo de datos incorporado. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

FLOAT

FLOAT

►►—FLOAT—(*—expresión-numérica—*)—◄◄

El esquema es SYSIBM.

La función FLOAT devuelve una representación de coma flotante de un número.

FLOAT es sinónimo de DOUBLE. Consulte el apartado “DOUBLE” en la página 329 para obtener detalles.

FLOOR

►►—FLOOR—(*—expresión—*)——————►►

El esquema es SYSFUN.

Devuelve el valor del entero más grande que es menor o igual que el argumento.

El argumento puede ser de cualquier tipo numérico incorporado. Si el argumento tiene el tipo DECIMAL o REAL, se convierte a un número de coma flotante de precisión doble para que lo procese la función. Si el argumento tiene el tipo SMALLINT, INTEGER o BIGINT, se devuelve el valor del argumento.

El resultado de la función es:

- SMALLINT si el argumento es SMALLINT
- INTEGER si el argumento es INTEGER
- BIGINT si el argumento es BIGINT
- DOUBLE si el argumento es DOUBLE, DECIMAL o REAL. Los valores decimales con más de 15 dígitos a la izquierda del decimal no devolverán el valor entero deseado debido a que pierden precisión en la conversión a DOUBLE.

El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

GETHINT

►—GETHINT—(*datos-cifrados*)—►

El esquema es SYSIBM.

La función GETHINT devolverá la indicación de contraseña si se encuentra alguna en *datos-cifrados*. Una indicación de contraseña es una expresión que ayuda a los propietarios de datos a recordar las contraseñas (por ejemplo, 'Océano' como indicación para recordar 'Pacífico').

datos-cifrados

Una expresión que devuelve un valor CHAR FOR BIT DATA o VARCHAR FOR BIT DATA que es una serie de datos cifrada completa. La serie de datos se tiene que haber cifrado utilizando la función ENCRYPT (SQLSTATE 428FE).

El resultado de la función es VARCHAR(32). El resultado puede ser nulo; si la función ENCRYPT no ha añadido el parámetro de indicación a los *datos-cifrados* o el primer argumento es nulo, el resultado será el valor nulo.

Consulte también el apartado "DECRYPT_BIN y DECRYPT_CHAR" en la página 314 y el apartado "ENCRYPT" en la página 331 para obtener información adicional sobre la utilización de esta función.

Ejemplo:

En este ejemplo se almacena la indicación 'Océano' para ayudar al usuario a recordar la contraseña de cifrado 'Pacífico'.

```
INSERT INTO EMP (SSN) VALUES ENCRYPT('289-46-8832', 'Pacífico','Océano');
SELECT GETHINT(SSN)
FROM EMP;
```

El valor devuelto es 'Océano'.

GENERATE_UNIQUE

►—GENERATE_UNIQUE—(—)—►

El esquema es SYSIBM.

La función GENERATE_UNIQUE devuelve una serie de caracteres de datos de bits de 13 bytes de longitud (CHAR(13) FOR BIT DATA) que es exclusiva comparada con cualquier otra ejecución de la misma función.³⁶ La función se define como no determinista.

No hay ningún argumento para esta función (se han de especificar los paréntesis vacíos).

El resultado de la función es un valor exclusivo que incluye el formato interno de la Hora universal coordinada (UTC) y el número de partición en la que se ha procesado la función. El resultado no puede ser nulo.

El resultado de esta función se puede utilizar para proporcionar valores exclusivos en una tabla. Cada valor sucesivo será mayor que el valor anterior, proporcionando una secuencia que se puede utilizar en una tabla. El valor incluye el número de partición en el que se ha ejecutado la función para que una tabla particionada en múltiples particiones también tenga valores exclusivos en algunas secuencias. La secuencia se basa en la hora en que se ha ejecutado la función.

Esta función difiere de la utilización del registro especial CURRENT TIMESTAMP en que se genera un valor exclusivo para cada fila de una sentencia de inserción de múltiples filas o en una sentencia de inserción con una selección completa.

El valor de indicación de la hora que forma parte del resultado de esta función puede determinarse utilizando la función escalar TIMESTAMP con el resultado de GENERATE_UNIQUE como argumento.

Ejemplos:

- Cree una tabla que incluya una columna que sea exclusiva para cada fila. Llene esta columna utilizando la función GENERATE_UNIQUE. Tenga en cuenta que la columna UNIQUE_ID tiene especificado "FOR BIT DATA" para identificar la columna como una serie de caracteres de datos de bits.

36. Se utiliza el reloj del sistema para generar la indicación de la Hora Universal Coordinada (UTC) junto con el número de partición en la que se ejecuta la función. Los ajustes que retrasan el reloj del sistema real podrían generar valores duplicados.

GENERATE_UNIQUE

```
CREATE TABLE EMP_UPDATE  
(UNIQUE_ID CHAR(13) FOR BIT DATA,  
EMPNO CHAR(6),  
TEXT VARCHAR(1000))
```

```
INSERT INTO EMP_UPDATE  
VALUES (GENERATE_UNIQUE(), '000020', 'Actualizar entrada...'),  
(GENERATE_UNIQUE(), '000050', 'Actualizar entrada...')
```

Esta tabla tendrá un identificador exclusivo para cada fila siempre que la columna UNIQUE_ID se establezca siempre utilizando GENERATE_UNIQUE. Esto se puede realizar introduciendo un desencadenante en la tabla.

```
CREATE TRIGGER EMP_UPDATE_UNIQUE  
NO CASCADE BEFORE INSERT ON EMP_UPDATE  
REFERENCING NEW AS NEW_UPD  
FOR EACH ROW MODE DB2SQL  
SET NEW_UPD.UNIQUE_ID = GENERATE_UNIQUE()
```

Con este desencadenante definido, la sentencia INSERT anterior se emitiría sin la primera columna, tal como se indica a continuación.

```
INSERT INTO EMP_UPDATE (EMPNO, TEXT)  
VALUES ('000020', 'Actualizar entrada 1...'),  
( '000050', 'Actualizar entrada 2...')
```

Puede devolverse la indicación de la hora (en UTC) para el momento en que se ha añadido una fila a EMP_UPDATE utilizando:

```
SELECT TIMESTAMP (UNIQUE_ID), EMPNO, TEXT FROM EMP_UPDATE
```

Por lo tanto, no hay necesidad de tener una columna de indicación de la hora en la tabla para registrar el momento en que se ha insertado una fila.

GRAPHIC

►►—GRAPHIC—(—*expresión-gráfica*—
,—*entero*—)—

El esquema es SYSIBM.

La función GRAPHIC devuelve una representación GRAPHIC de un tipo de serie gráfica.

expresión-gráfica

Una *expresión* que devuelve un valor que es una serie gráfica.

entero

Un valor entero que especifica el atributo de longitud del tipo de datos GRAPHIC resultante. El valor debe estar entre 1 y 127. Si no se especifica *entero*, la longitud del resultado es la misma que la longitud del primer argumento.

El resultado de la función es GRAPHIC. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

HEX

HEX

►—HEX—(*—expresión—*)—►

El esquema es SYSIBM.

La función HEX devuelve una representación hexadecimal de un valor como una serie de caracteres.

El argumento puede ser una expresión que sea un valor de cualquier tipo de datos incorporado, con una longitud máxima de 16.336 bytes.

El resultado de la función es una serie de caracteres. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

La página de códigos es la página de códigos de la base de datos.

El resultado es una serie de dígitos hexadecimales. Los dos primeros bytes representan el primer byte del argumento, los dos siguientes el segundo byte del argumento, etcétera. Si el argumento es un valor de indicación de fecha y hora o un valor numérico el resultado es la representación hexadecimal del formato interno del argumento. La representación hexadecimal que se devuelve puede ser diferente según el servidor de aplicaciones donde se ejecuta la función. Los casos en que las diferencias pueden ser evidentes son:

- Los argumentos de serie de caracteres cuando se ejecuta la función HEX en un cliente ASCII con un servidor EBCDIC o en un cliente EBCDIC con un servidor ASCII.
- Los argumentos numéricos (en algunos casos) cuando se ejecuta la función HEX donde los sistemas cliente y servidor tienen distintas clasificaciones de bytes para los valores numéricos.

El tipo y la longitud del resultado varían basándose en el tipo y la longitud de los argumentos de serie de caracteres.

- Serie de caracteres
 - Longitud fija no mayor que 127
 - El resultado es una serie de caracteres de longitud fija el doble de la longitud definida del argumento.
 - Longitud fija mayor que 127
 - El resultado es una serie de caracteres de longitud variable el doble de la longitud definida del argumento.
 - Longitud variable

- El resultado es una serie de caracteres de longitud variable con una longitud máxima el doble de la longitud máxima definida del argumento.
- Serie gráfica
 - Longitud fija no mayor que 63
 - El resultado es una serie de caracteres de longitud fija cuatro veces la longitud definida del argumento.
- Longitud fija mayor que 63
 - El resultado es una serie de caracteres de longitud variable cuatro veces la longitud definida del argumento.
- Longitud variable
 - El resultado es una serie de caracteres de longitud variable con una longitud máxima cuatro veces la longitud máxima definida del argumento.

Ejemplos:

Suponga que utiliza un servidor de aplicaciones DB2 para AIX en los ejemplos siguientes.

- Utilizando la tabla DEPARTMENT establezca la variable del lenguaje principal HEX_MGRNO (char(12)) en la representación hexadecimal del número del director (MGRNO) para el departamento 'PLANNING' (DEPTNAME).

```

SELECT HEX(MGRNO)
  INTO :HEX_MGRNO
  FROM DEPARTMENT
 WHERE DEPTNAME = 'PLANNING'

```

HEX_MGRNO se establecerá en '303030303230' cuando se utilice la tabla de ejemplo (el valor de caracteres es '000020').

- Suponga que COL_1 es una columna con un tipo de datos de char(1) y un valor de 'B'. La representación hexadecimal de la letra 'B' es X'42'. HEX(COL_1) devuelve una serie de dos caracteres '42'.
- Suponga que COL_3 es una columna con un tipo de datos de decimal(6,2) y un valor de 40,1. Una serie de ocho caracteres '0004010C' es el resultado de aplicar la función HEX a la representación interna del valor decimal 40,1.

HOUR

HOUR

►—HOUR—(*—expresión—*)—◄

El esquema es SYSIBM.

La función HOUR devuelve la parte correspondiente a la hora de un valor.

El argumento debe ser una hora, una indicación de fecha y hora, una duración de hora, una duración de indicación de fecha y hora o una representación de serie de caracteres válida de una hora o de una fecha y hora que no sea CLOB ni LONG VARCHAR.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Las demás reglas dependen del tipo de datos del argumento:

- Si el argumento es una hora, una indicación de fecha y hora o una representación de serie válida de una hora o de una fecha y hora:
 - El resultado es la parte correspondiente a la hora del valor, que es un entero entre 0 y 24.
- Si el argumento es una duración de hora o una duración de indicación de fecha y hora:
 - El resultado es la parte correspondiente a la hora del valor, que es un entero entre -99 y 99. El resultado que no es cero tiene el mismo signo que el argumento.

Ejemplo:

Utilizando la tabla de ejemplo CL_SCHED, seleccione todas las clases que empiezan por la tarde.

```
SELECT * FROM CL_SCHED  
WHERE HOUR(STARTING) BETWEEN 12 AND 17
```

IDENTITY_VAL_LOCAL

►►—IDENTITY_VAL_LOCAL—(—)—►►

El esquema es SYSIBM.

La función IDENTITY_VAL_LOCAL es una función no determinista que devuelve el valor asignado más recientemente de una columna de identidad, donde la asignación se ha producido como resultado de una sentencia de inserción (INSERT) de fila individual utilizando una cláusula VALUES. La función no tiene parámetros de entrada.

El resultado es un DECIMAL(31,0), independientemente del tipo de datos real de la columna de identidad correspondiente.

El valor devuelto por la función es el valor asignado a la columna de identidad de la tabla identificada en la sentencia INSERT de fila individual más reciente. La sentencia INSERT debe ejecutarse utilizando una cláusula VALUES en una tabla que contenga una columna de identidad. Asimismo, la sentencia INSERT debe emitirse al mismo nivel³⁷ (es decir, el valor está disponible localmente en el nivel al que se ha asignado, hasta que se sustituye por el siguiente valor asignado).

El valor asignado es un valor proporcionado por el usuario (si la columna de identidad está definida como GENERATED BY DEFAULT) o un valor de identidad generado por DB2.

La función devuelve un valor nulo en las situaciones siguientes:

- Cuando no se ha emitido una sentencia INSERT de fila individual con una cláusula VALUES en el nivel de proceso actual para una tabla que contiene una columna de identidad.
- Cuando se ha producido una operación COMMIT o ROLLBACK de una unidad de trabajo desde que se emitió la sentencia INSERT más reciente que asignó un valor³⁸.

El resultado de la función no queda afectado por lo siguiente:

- Una sentencia INSERT de fila individual con una cláusula VALUES para una tabla sin columna de identidad.
- Una sentencia INSERT de múltiples filas con una cláusula VALUES.

37. Se inicia un nivel nuevo cada vez que se invoca un desencadenante, una función o un procedimiento almacenado.

38. A menos que se desactive la confirmación automática, las interfaces que realizan confirmaciones automáticas después de cada sentencia devolverán un valor nulo cuando se invoca la función en sentencias independientes.

IDENTITY_VAL_LOCAL

- Una sentencia INSERT con una selección completa.
- Una sentencia ROLLBACK TO SAVEPOINT.

Notas:

- Las expresiones de la cláusula VALUES de una sentencia INSERT se evalúan antes que las asignaciones para las columnas de destino de la sentencia INSERT. Por consiguiente, una invocación de una función IDENTITY_VAL_LOCAL en la cláusula VALUES de una sentencia INSERT utilizará el valor asignado más recientemente de una columna de identidad de una sentencia INSERT anterior. La función devuelve el valor nulo si no se ha ejecutado ninguna sentencia INSERT de fila individual anterior con una cláusula VALUES para una tabla que contiene una columna de identidad dentro del mismo nivel que la función IDENTITY_VAL_LOCAL.
- El valor de columna de identidad de la tabla para la que se define el desencadenante puede determinarse dentro de un desencadenante, haciendo referencia a la variable de transición desencadenante para la columna de identidad.
- El resultado de la invocación de la función IDENTITY_VAL_LOCAL desde dentro de la condición desencadenante de un desencadenante de inserción es un valor nulo.
- Es posible que existan múltiples desencadenantes de inserción anteriores o posteriores para una tabla. En este caso, cada desencadenante se procesa por separado y los valores de identidad asignados por una acción desencadenada no están disponibles para las demás acciones desencadenadas utilizando la función IDENTITY_VAL_LOCAL. Esto es válido incluso aunque las múltiples acciones desencadenadas estén definidas conceptualmente al mismo nivel.
- Generalmente no es recomendable utilizar la función IDENTITY_VAL_LOCAL en el cuerpo de un desencadenante anterior (before) de inserción. El resultado de la invocación de la función IDENTITY_VAL_LOCAL desde dentro de la acción desencadenada de un desencadenante de inserción anterior es el valor nulo. El valor de la columna de identidad de la tabla para la que se ha definido el desencadenante no se puede obtener invocando la función IDENTITY_VAL_LOCAL en la acción desencadenada de un desencadenante de inserción anterior. Sin embargo, el valor de la columna de identidad puede obtenerse en la acción desencadenada, haciendo referencia a la variable de transición desencadenante para la columna de identidad.
- El resultado de la invocación de la función IDENTITY_VAL_LOCAL desde dentro de la acción desencadenada de un desencadenante posterior (after) de inserción³⁹ es el valor asignado a una columna de identidad de la tabla identificada en la sentencia INSERT de fila individual más reciente

39. Esto se aplica a los desencadenantes posteriores (after) de inserción FOR EACH ROW y FOR EACH STATEMENT.

invocada en la misma acción desencadenada que tenía una cláusula VALUES para una tabla que contenía una columna de identidad. Si, antes de la invocación de la función IDENTITY_VAL_LOCAL, no se ha ejecutado una sentencia INSERT de fila individual con una cláusula VALUES para una tabla que contiene una columna de identidad dentro de la misma acción desencadenada, la función devolverá un valor nulo.

- Dado que los resultados de la función IDENTITY_VAL_LOCAL no son deterministas, el resultado de una invocación de la función IDENTITY_VAL_LOCAL dentro de la sentencia SELECT de un cursor puede variar para cada sentencia FETCH.
- El valor asignado es el valor realmente asignado a la columna de identidad (es decir, el valor que se devolverá en una sentencia SELECT subsiguiente). Este valor no es necesariamente el valor proporcionado en la cláusula VALUES de la sentencia INSERT o un valor generado por DB2. El valor asignado puede ser un valor especificado en una sentencia de variable de transición SET, dentro del cuerpo de un desencadenante de inserción anterior, para una variable de transición desencadenante asociada con la columna de identidad.
- El valor devuelto por la función es imprevisible después de una operación INSERT de fila individual anómala con una cláusula VALUES en una tabla con una columna de identidad. El valor puede ser el valor que se devolvería de la función si ésta se hubiera invocado antes del INSERT anómalo o puede ser el valor que se asignaría si el INSERT hubiera sido satisfactorio. El valor real devuelto depende del punto de anomalía y, por consiguiente, es imprevisible.

Ejemplos:

Ejemplo 1: Establezca la variable IVAR en el valor asignado a la columna de identidad de la tabla EMPLOYEE. Si esta inserción es la primera en la tabla EMPLOYEE, IVAR tendrá un valor de 1.

```
CREATE TABLE EMPLOYEE
  (EMPNO   INTEGER GENERATED ALWAYS AS IDENTITY,
   NAME    CHAR(30),
   SALARY  DECIMAL(5,2),
   DEPTNO  SMALLINT)
```

Ejemplo 2: Una función IDENTITY_VAL_LOCAL invocada en una sentencia INSERT devuelve el valor asociado con la sentencia INSERT de fila individual anterior, con una cláusula VALUES para una tabla con una columna de identidad. Para este ejemplo suponga que existen dos tablas, T1 y T2. T1 y T2 tienen una columna de identidad llamada C1. DB2 genera valores en secuencia, empezando por 1, para la columna C1 de la tabla T1 y valores en secuencia, empezando por 10, para la columna C1 de la tabla T2.

IDENTITY_VAL_LOCAL

```
CREATE TABLE T1
  (C1 INTEGER GENERATED ALWAYS AS IDENTITY,
   C2 INTEGER),
CREATE TABLE T2
  (C1 DECIMAL(15,0) GENERATED BY DEFAULT AS IDENTITY
   (START WITH 10),
   C2 INTEGER),
INSERT INTO T1 (C2) VALUES (5),
INSERT INTO T1 (C2) VALUES (6),
SELECT * FROM T1
```

lo que da un resultado de:

C1	C2
-----	-----
1	5
2	6

y ahora, declarando la función para la variable IVAR:

```
VALUES IDENTITY_VAL_LOCAL() INTO :IVAR
```

En este punto, la función `IDENTITY_VAL_LOCAL` devolverá un valor de 2 en `IVAR`, porque ése es el valor que DB2 ha asignado más recientemente. La sentencia `INSERT` siguiente inserta una fila individual en `T2`, donde la columna `C2` obtiene un valor de 2 de la función `IDENTITY_VAL_LOCAL`.

```
INSERT INTO T2 (C2) VALUES (IDENTITY_VAL_LOCAL());
SELECT * FROM T2
WHERE C1 = DECIMAL(IDENTITY_VAL_LOCAL(),15,0)
```

lo que devuelve un resultado de:

C1	C2
-----	-----
10.	2

La invocación de la función `IDENTITY_VAL_LOCAL` después de esta inserción produce un valor de 10, que es el valor generado por DB2 para la columna `C1` de `T2`.

En un entorno anidado que incluya un desencadenante, utilice la función `IDENTITY_VAL_LOCAL` para recuperar el valor de identidad asignado en un nivel determinado, incluso aunque puedan haber valores de identidad asignados en niveles inferiores. Suponga que existen tres tablas, `EMPLOYEE`, `EMP_ACT` y `ACCT_LOG`. Hay un desencadenante de inserción posterior definido en `EMPLOYEE` que produce inserciones adicionales en las tablas `EMP_ACT` y `ACCT_LOG`.

```
CREATE TABLE EMPLOYEE
  (EMPNO SMALLINT GENERATED ALWAYS AS IDENTITY (START WITH 1000),
   NAME CHAR(30),
   SALARY DECIMAL(5,2),
   DEPTNO SMALLINT);
```

```

CREATE TABLE EMP_ACT
  (ACNT_NUM SMALLINT GENERATED ALWAYS AS IDENTITY (START WITH 1),
  EMPNO SMALLINT);

CREATE TABLE ACCT_LOG
  (ID SMALLINT GENERATED ALWAYS AS IDENTITY (START WITH 100),
  ACNT_NUM SMALLINT,
  EMPNO SMALLINT);

CREATE TRIGGER NEW_HIRE
  AFTER INSERT ON EMPLOYEE
  REFERENCING NEW AS NEW_EMP
  FOR EACH ROW MODE DB2SQL
  BEGIN ATOMIC
    INSERT INTO EMP_ACT (EMPNO)
      VALUES (NEW_EMP.EMPNO);
    INSERT INTO ACCT_LOG (ACNT_NUM EMPNO)
      VALUES (IDENTITY_VAL_LOCAL(), NEW_EMP.EMPNO);
  END

```

La primera sentencia INSERT desencadenada inserta una fila en la tabla EMP_ACT. Esta sentencia INSERT utiliza una variable de transición desencadenante para la columna EMPNO de la tabla EMPLOYEE, para indicar que el valor de identidad para la columna EMPNO de la tabla EMPLOYEE debe copiarse en la columna EMPNO de la tabla EMP_ACT. La función IDENTITY_VAL_LOCAL no se ha podido utilizar para obtener el valor asignado a la columna EMPNO de la tabla EMPLOYEE. Esto se debe a que no se ha emitido una sentencia INSERT en este nivel del anidamiento y, por lo tanto, si se hubiera invocado la función IDENTITY_VAL_LOCAL en la cláusula VALUES de INSERT para EMP_ACT, se hubiera devuelto un valor nulo. Esta sentencia INSERT para la tabla EMP_ACT también hace que se genere un nuevo valor de columna de identidad para la columna ACNT_NUM.

Una segunda sentencia INSERT desencadenada inserta una fila en la tabla ACCT_LOG. Esta sentencia invoca la función IDENTITY_VAL_LOCAL para indicar que el valor de identidad asignado a la columna ACNT_NUM de la tabla EMP_ACT en la sentencia INSERT anterior de la acción desencadenada debe copiarse en la columna ACNT_NUM de la tabla ACCT_LOG. A la columna EMPNO se le asigna el mismo valor que a la columna EMPNO de la tabla EMPLOYEE.

Desde la aplicación que realiza la invocación (es decir, el nivel en el que se emite INSERT en EMPLOYEE), establezca la variable IVAR en el valor asignado a la columna EMPNO de la tabla EMPLOYEE por la sentencia INSERT original.

```

INSERT INTO EMPLOYEE (NAME, SALARY, DEPTNO)
VALUES ('Rupert', 989.99, 50);

```

IDENTITY_VAL_LOCAL

El contenido de las tres tablas después de procesar la sentencia INSERT original y todas las acciones desencadenadas es:

```
SELECT EMPNO, SUBSTR(NAME,10) AS NAME, SALARY, DEPTNO
FROM EMPLOYEE;
```

EMPNO	NAME	SALARY	DEPTNO
1000	Rupert	989.99	50

```
SELECT ACNT_NUM, EMPNO
FROM EMP_ACT;
```

ACNT_NUM	EMPNO
1	1000

```
SELECT * FROM ACCT_LOG;
```

ID	ACNT_NUM	EMPNO
100	1	1000

El resultado de la función IDENTITY_VAL_LOCAL es el valor asignado más recientemente para una columna de identidad en el mismo nivel de anidamiento. Después de procesar la sentencia INSERT original y todas las acciones desencadenadas, la función IDENTITY_VAL_LOCAL devuelve un valor de 1000, porque éste es el valor asignado a la columna EMPNO de la tabla EMPLOYEE. La sentencia VALUES siguiente hace que IVAR se establezca en 1000. La inserción en la tabla EMP_ACT (que se ha producido después de la inserción en la tabla EMPLOYEE y a un nivel de anidamiento inferior) no tiene ningún efecto en lo que devuelve esta invocación de la función IDENTITY_VAL_LOCAL.

```
VALUES IDENTITY_VAL_LOCAL() INTO :IVAR;
```

INSERT

►►—INSERT—(—*expresión1*—,—*expresión2*—,—*expresión3*—,—*expresión4*—)—►►

El esquema es SYSFUN.

Devuelve una serie en la que se han suprimido *expresión3* bytes de *expresión1*, empezando en la *expresión2* y donde se ha insertado la *expresión4* en la *expresión1* empezando en la *expresión2*. Si la longitud de la serie del resultado excede el máximo para el tipo de retorno, se produce un error (SQLSTATE 38552).

El primer argumento es una serie de caracteres o un tipo de serie binaria. El segundo argumento y el tercero deben ser un valor numérico con un tipo de datos SMALLINT o INTEGER. Si el primer argumento es una serie de caracteres, entonces el cuarto argumento también ha de ser una serie de caracteres. Si el primer argumento es una serie binaria, entonces el cuarto argumento también ha de ser una serie binaria. Para un VARCHAR, la longitud máxima es 4.000 bytes, y para un CLOB o serie binaria, la longitud máxima es 1.048.576 bytes. Para el primer y cuarto argumentos, CHAR se convierte a VARCHAR y LONG VARCHAR a CLOB(1M); para el segundo y tercer argumentos, SMALLINT se convierte a INTEGER para que lo procese la función.

El resultado se basa en los tipos de argumentos tal como sigue:

- VARCHAR(4000) si el primer argumento y el cuarto son VARCHAR (no exceden de 4.000 bytes) o CHAR
- CLOB(1M) si el primer argumento o el cuarto es CLOB o LONG VARCHAR
- BLOB(1M) si el primer argumento y el cuarto son BLOB.

El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

Ejemplo:

- Suprima un carácter de la palabra 'DINING' e inserte 'VID', empezando en el tercer carácter.

```
VALUES CHAR(INSERT('DINING', 3, 1, 'VID'), 10)
```

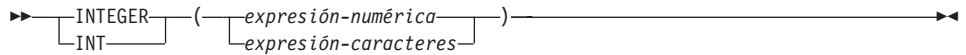
Este ejemplo devuelve lo siguiente:

```
1
-----
DIVIDING
```

INSERT

Tal como se menciona, el resultado de la función INSERT es VARCHAR(4000). En el ejemplo anterior, se ha utilizado la función CHAR para limitar la salida de INSERT a 10 bytes. La ubicación inicial de una serie en particular se puede encontrar utilizando LOCATE. Consulte el apartado "LOCATE" en la página 362 para obtener más información.

INTEGER



El esquema es SYSIBM.

La función INTEGER devuelve una representación entera de 64 bits de un número o serie de caracteres, en forma de constante entera.

expresión-numérica

Una expresión que devuelve un valor de cualquier tipo de datos numérico incorporado.

Si el argumento es una *expresión-numérica*, el resultado es el mismo número que sería si el argumento se asignase a una columna o variable de enteros grandes. Si la parte correspondiente a los enteros del argumento no está dentro del rango de enteros, se produce un error. La parte correspondiente a los decimales del argumento se trunca si está presente.

expresión-caracteres

Una expresión que devuelve un valor de serie de caracteres de longitud no mayor que la longitud máxima de una constante de caracteres. Se eliminan los blancos iniciales y de cola y la serie resultante debe ajustarse a las reglas para la formación de una constante de enteros SQL (SQLSTATE 22018). La serie de caracteres no puede ser una serie larga.

Si el argumento es una *expresión-caracteres*, el resultado es el mismo número que sería si la constante de enteros correspondiente se asignase a una columna o variable de enteros grandes.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplos:

- Utilizando la tabla EMPLOYEE, seleccione una lista que contenga el salario (SALARY) dividido por el nivel de formación (EDLEVEL). Trunque cualquier decimal en el cálculo. La lista también debe contener los valores utilizados en el cálculo y el número de empleado (EMPNO). La lista debe estar en orden descendente del valor calculado.

```
SELECT INTEGER (SALARY / EDLEVEL), SALARY, EDLEVEL, EMPNO
FROM EMPLOYEE
```

ORDER BY 1 DESC

- Utilizando la tabla EMPLOYEE, seleccione la columna EMPNO en el formato de enteros para procesarla más en la aplicación.

INTEGER

```
SELECT INTEGER(EMPNO) FROM EMPLOYEE
```


JULIAN_DAY

►►—JULIAN_DAY—(*—expresión—*)—►►

El esquema es SYSFUN.

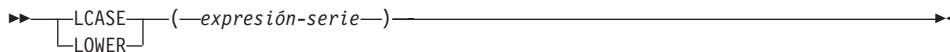
Devuelve un valor entero que representa el número de días desde el 1 de enero de 4712 A.C. (el inicio del calendario Juliano) hasta el valor de la fecha especificada en el argumento.

El argumento debe ser una fecha, una indicación de la hora o una representación de serie de caracteres válida de una fecha o indicación de la hora que no sea CLOB ni LONG VARCHAR.

El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

LCASE o LOWER

LCASE o LOWER



El esquema es SYSIBM. ⁴⁰

Las funciones LCASE o LOWER devuelve una serie en la cual todos los caracteres SBCS se han convertido a minúsculas (es decir, los caracteres de la A a la Z se convertirán en los caracteres de la a a la z, y los caracteres con signos diacríticos se convertirán a sus minúsculas equivalentes, si existen. Por ejemplo, en la página de códigos 850, É se correlaciona con é). Puesto que no se convierten todos los caracteres, LCASE(UCASE(*expresión-serie*)) no devuelve necesariamente el mismo resultado que LCASE(*expresión-serie*).

El argumento debe ser una expresión cuyo valor sea un tipo de datos CHAR o VARCHAR.

El resultado de la función tiene el mismo tipo de datos y el mismo atributo de longitud que el argumento. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Notas:

Esta función se ha ampliado para reconocer las propiedades de minúsculas y mayúsculas de un carácter Unicode. En una base de datos Unicode, todos los caracteres Unicode se convierten correctamente a minúsculas.

Ejemplo: Asegúrese de que los caracteres del valor de la columna JOB de la tabla EMPLOYEE se devuelven en minúsculas.

```
SELECT LCASE(JOB)
FROM EMPLOYEE WHERE EMPNO = '000020';
```

El resultado es el valor 'director'.

40. Continúa estando disponible la versión SYSFUN de esta función con el soporte para los argumentos LONG VARCHAR y CLOB. Vea "LCASE (esquema SYSFUN)" en la página 357 para obtener una descripción.

LCASE (esquema SYSFUN)

►►—LCASE—(—*expresión*—)—————►►

El esquema es SYSFUN.

Devuelve una serie en la que todos los caracteres de la A a la Z se han convertido en caracteres de la a a la z (caracteres con signos diacríticos no convertidos). Tenga en cuenta que, por lo tanto, LCASE(UCASE(serie)) no devolverá necesariamente el mismo resultado que LCASE(serie).

El argumento puede ser de cualquier tipo de serie de caracteres incorporado. Para un VARCHAR, la longitud máxima es 4.000 bytes y para un CLOB la longitud máxima es 1.048.576 bytes.

El resultado de la función es:

- VARCHAR(4000) si el argumento es VARCHAR (no excede de 4.000 bytes) o CHAR
- CLOB(1 M) si el argumento es CLOB o LONG VARCHAR

El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

LEFT

LEFT

▶▶—LEFT—(—*expresión1*—,—*expresión2*—)————▶▶

El esquema es SYSFUN.

Devuelve una serie que consta de los *expresión2* bytes situados más a la izquierda de *expresión1*. El valor *expresión1* se rellena con blancos por la derecha para que la subserie especificada de *expresión1* exista siempre.

El primer argumento es una serie de caracteres o un tipo de serie binaria. Para un VARCHAR, la longitud máxima es 4.000 bytes, y para un CLOB o serie binaria, la longitud máxima es 1.048.576 bytes. El segundo argumento debe ser del tipo de datos INTEGER o SMALLINT.

El resultado de la función es:

- VARCHAR(4000) si el argumento es VARCHAR (no excede de 4.000 bytes) o CHAR
- CLOB(1 M) si el argumento es CLOB o LONG VARCHAR
- BLOB(1 M) si el argumento es BLOB.

El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

LENGTH

►►—LENGTH—(—*expresión*—)—————►►

El esquema es SYSIBM.

La función LENGTH devuelve la longitud de un valor.

El argumento puede ser una expresión que devuelve un valor de cualquier tipo de datos incorporados.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

El resultado es la longitud del argumento. La longitud no incluye el byte indicador de nulo de los argumentos de columna que permiten valores nulos. La longitud de las series incluyen blancos pero no incluyen el campo de control de longitud de las series de longitud variable. La longitud de una serie de longitud variable es la longitud real, no la longitud máxima.

La longitud de una serie gráfica es el número de caracteres DBCS. La longitud de todos los demás valores es el número de bytes utilizados para representar el valor:

- 2 para entero pequeño
- 4 para entero grande
- $(p/2)+1$ para números decimales con la precisión p
- La longitud de la serie para series binarias
- La longitud de la serie para series de caracteres
- 4 para coma flotante de precisión simple
- 8 para coma flotante de precisión doble
- 4 para fecha
- 3 para hora
- 10 para fecha y hora

Ejemplos:

- Suponga que la variable del lenguaje principal ADDRESS es una serie de caracteres de longitud variable con un valor de '895 Don Mills Road'.

LENGTH (:ADDRESS)

Devuelve el valor 18.

LENGTH

- Suponga que `START_DATE` es una columna de tipo `DATE`.

`LENGTH(START_DATE)`

Devuelve el valor 4.

- Este ejemplo devuelve el valor 10.

`LENGTH(CHAR(START_DATE, EUR))`

LN

►►—LN—(*—expresión—*)——————▶◀

El esquema es SYSFUN.

Devuelve el logaritmo natural del argumento (igual que LOG).

El argumento puede ser de cualquier tipo de datos incorporado. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

LOCATE

LOCATE

►—LOCATE—(—*expresión1*—,—*expresión2*—,—*expresión3*—)—►

El esquema es SYSFUN.

Devuelve la posición inicial de la primera ocurrencia de *expresión1* en la *expresión2*. Si se especifica la *expresión3* opcional, indica la posición de los caracteres en *expresión2* en la que tiene que empezar la búsqueda. Si no se encuentra la *expresión1* en la *expresión2*, se devuelve el valor 0.

Si el primer argumento es una serie de caracteres, entonces el segundo ha de ser una serie de caracteres. Para un VARCHAR, la longitud máxima es 4.000 bytes y para un CLOB la longitud máxima es 1.048.576 bytes. Si el primer argumento es una serie binaria, entonces el segundo argumento debe ser una serie binaria con una longitud máxima de 1.048.576 bytes. El tercer argumento debe ser INTEGER o SMALLINT.

El resultado de la función es INTEGER. El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

Ejemplo:

- Busque la ubicación de la letra 'N' (primera ocurrencia) en la palabra 'DINING'.

VALUES LOCATE ('N', 'DINING')

Este ejemplo devuelve lo siguiente:

```
1  
-----  
3
```


LOG

►►—LOG—(*—expresión—*)—◄◄

El esquema es SYSFUN.

Devuelve el logaritmo natural del argumento (igual que LN).

El argumento puede ser de cualquier tipo de datos incorporado. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

LOG10

LOG10

► LOG10(*—expresión—*) ◄

El esquema es SYSFUN.

Devuelve el logaritmo de base 10 del argumento.

El argumento puede ser de cualquier tipo numérico incorporado. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

LONG_VARCHAR

►►—LONG_VARCHAR—(—*expresión-serie-caracteres*—)—————►►

El esquema es SYSIBM.

La función LONG_VARCHAR devuelve una representación LONG VARCHAR de un tipo de datos de serie de caracteres.

expresión-serie-caracteres

Una *expresión* que devuelve un valor que es una serie de caracteres con una longitud máxima de 32 700 bytes.

El resultado de la función es LONG VARCHAR. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

LONG_VARGRAPHIC

LONG_VARGRAPHIC

►►—LONG_VARGRAPHIC—(*—expresión-gráfica—*)——————►◄

El esquema es SYSIBM.

La función LONG_VARGRAPHIC devuelve una representación LONG VARGRAPHIC de una serie de caracteres de doble byte.

expresión-gráfica

Una *expresión* que devuelve un valor que es una serie gráfica con una longitud máxima de 16 350 caracteres de doble byte.

El resultado de la función es LONG VARGRAPHIC. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

LTRIM

►—LTRIM—(*—expresión-serie—*)—◄

El esquema es SYSIBM. ⁴¹

La función LTRIM elimina los blancos del principio de la *expresión-serie*.

El argumento puede ser un tipo de datos CHAR, VARCHAR, GRAPHIC o VARGRAPHIC.

- Si el argumento es una serie gráfica de una base de datos DBCS o EUC, se eliminan los blancos de doble byte iniciales.
- Si el argumento es una serie gráfica de una base de datos Unicode, se eliminan los blancos UCS-2 iniciales.
- De lo contrario, se eliminan los blancos de un solo byte iniciales.

El tipo de datos del resultado de la función es:

- VARCHAR si el tipo de datos de *expresión-serie* es VARCHAR o CHAR
- VARGRAPHIC si el tipo de datos de *expresión-serie* es VARGRAPHIC o GRAPHIC

El parámetro de longitud del tipo devuelto es el mismo que el parámetro de longitud del tipo de datos del argumento.

La longitud real del resultado para las series de caracteres es la longitud de *expresión-serie* menos el número de bytes eliminados debido a caracteres en blanco. La longitud real del resultado para series gráficas es la longitud (en número de caracteres de doble byte) de *expresión-serie* menos el número de caracteres en blanco de doble byte eliminados. Si se eliminan todos los caracteres, el resultado es una serie vacía, de longitud variable (la longitud es cero).

Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplo: Suponga que la variable del lenguaje principal HELLO está definida como CHAR(9) y tiene el valor 'Hola'.

```
VALUES LTRIM(:HELLO)
```

El resultado es 'Hola'.

41. Continúa estando disponible la versión SYSFUN de esta función con el soporte para los argumentos LONG VARCHAR y CLOB. Vea "LTRIM (esquema SYSFUN)" en la página 368 para obtener una descripción.

LTRIM (esquema SYSFUN)

LTRIM (esquema SYSFUN)

►—LTRIM—(—*expresión*—)—————►

El esquema es SYSFUN.

Devuelve los caracteres del argumento con los blancos iniciales eliminados.

El argumento puede ser de cualquier tipo de serie de caracteres incorporado. Para un VARCHAR, la longitud máxima es 4.000 bytes y para un CLOB la longitud máxima es 1.048.576 bytes.

El resultado de la función es:

- VARCHAR(4000) si el argumento es VARCHAR (no excede de 4.000 bytes) o CHAR
- CLOB(1 M) si el argumento es CLOB o LONG VARCHAR.

El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

MICROSECOND

►►—MICROSECOND—(—*expresión*—)—————►►

El esquema es SYSIBM.

La función MICROSECOND devuelve la parte correspondiente a los microsegundos de un valor.

El argumento debe ser una indicación de la hora, duración de indicación de la hora o una representación de serie de caracteres válida de una indicación de la hora que no sea CLOB ni LONG VARCHAR.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Las demás reglas dependen del tipo de datos del argumento:

- Si el argumento es una indicación de la hora o una representación de serie válida de una indicación de la hora:
 - El entero está en el rango de 0 a 999 999.
- Si el argumento es una duración:
 - El resultado refleja la parte correspondiente a los microsegundos del valor que es un entero entre –999 999 y 999 999. El resultado que no es cero tiene el mismo signo que el argumento.

Ejemplo:

- Suponga que una tabla TABLEA contiene dos columnas, TS1 y TS2, del tipo TIMESTAMP. Seleccione todas las filas cuya parte correspondiente a los microsegundos de TS1 no sea cero y las partes correspondientes a los segundos de TS1 y TS2 sean idénticas.

```
SELECT * FROM TABLEA
WHERE MICROSECOND(TS1) <> 0 AND
SECOND(TS1) = SECOND(TS2)
```

MIDNIGHT_SECONDS

MIDNIGHT_SECONDS

►—MIDNIGHT_SECONDS—(*—expresión—*)—◄

El esquema es SYSFUN.

Devuelve un valor entero en el rango de 0 a 86 400 que representa el número de segundos entre medianoche y el valor de la hora especificado en el argumento.

El argumento debe ser una hora, una indicación de la hora o una representación de serie de caracteres válida de una hora o indicación de la hora que no sea CLOB ni LONG VARCHAR.

El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplo:

- Encuentre el número de segundos entre la medianoche y 00:10:10 y la medianoche y 13:10:10.

```
VALUES (MIDNIGHT_SECONDS('00:10:10'), MIDNIGHT_SECONDS('13:10:10'))
```

Este ejemplo devuelve lo siguiente:

1	2
-----	-----
610	47410

Puesto que un minuto es 60 segundos, hay 610 segundos entre la medianoche y la hora especificada. Es lo mismo para el segundo ejemplo. Hay 3600 segundos en una hora y 60 segundos en un minuto, dando como resultado 47410 segundos entre la hora especificada y la medianoche.

- Encuentre el número de segundos entre la medianoche y 24:00:00, y la medianoche y 00:00:00.

```
VALUES (MIDNIGHT_SECONDS('24:00:00'), MIDNIGHT_SECONDS('00:00:00'))
```

Este ejemplo devuelve lo siguiente:

1	2
-----	-----
86400	0

Observe que estos dos valores representan el mismo momento en el tiempo, pero devuelven distintos valores MIDNIGHT_SECONDS.

MINUTE

►►—MINUTE—(—*expresión*—)—————►►

El esquema es SYSIBM.

La función MINUTE devuelve la parte correspondiente a los minutos de un valor.

El argumento debe ser una hora, una indicación de fecha y hora, una duración de hora, una duración de indicación de fecha y hora o una representación de serie de caracteres válida de una hora o de una fecha y hora que no sea CLOB ni LONG VARCHAR.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Las demás reglas dependen del tipo de datos del argumento:

- Si el argumento es una hora, una indicación de fecha y hora o una representación de serie válida de una hora o de una fecha y hora:
 - El resultado es la parte correspondiente a los minutos de un valor, que es un entero entre 0 y 59.
- Si el argumento es una duración de hora o una duración de indicación de fecha y hora:
 - El resultado es la parte correspondiente a los minutos del valor, que es un entero entre -99 y 99. El resultado que no es cero tiene el mismo signo que el argumento.

Ejemplo:

- Utilizando la tabla de ejemplo CL_SCHED, seleccione todas las clases con una duración inferior a 50 minutos.

```
SELECT * FROM CL_SCHED
WHERE HOUR(ENDING - STARTING) = 0 AND
MINUTE(ENDING - STARTING) < 50
```

MOD

MOD

►► MOD (—*expresión*—, —*expresión*—) ◀◀

El esquema es SYSFUN.

Devuelve el resto del primer argumento dividido por el segundo argumento. El resultado sólo es negativo si el primer argumento es negativo.

El resultado de la función es:

- SMALLINT si ambos argumentos son SMALLINT
- INTEGER si un argumento es INTEGER y el otro es INTEGER o SMALLINT
- BIGINT si un argumento es BIGINT y el otro argumento es BIGINT, INTEGER o SMALLINT.

El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

MONTH

►►—MONTH—(—expresión—)—————►►

El esquema es SYSIBM.

La función MONTH devuelve la parte correspondiente al mes de un valor.

El argumento debe ser una fecha, una indicación de la hora, una duración de fecha, una duración de indicación de la hora o una representación de serie de caracteres válida de una fecha o indicación de la hora que no sea CLOB ni LONG VARCHAR.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Las demás reglas dependen del tipo de datos del argumento:

- Si el argumento es una fecha, una indicación de la hora o una representación de serie válida de una fecha o indicación de la hora:
 - El resultado es la parte correspondiente al mes del valor, que es un entero entre 1 y 12.
- Si el argumento es una duración de fecha o duración de indicación de la hora:
 - El resultado es la parte correspondiente al mes del valor, que es un entero entre -99 y 99. El resultado que no es cero tiene el mismo signo que el argumento.

Ejemplo:

- Seleccione todas las filas de la tabla EMPLOYEE de las personas que han nacido (BIRTHDATE) en diciembre (DECEMBER).

```
SELECT * FROM EMPLOYEE
WHERE MONTH(BIRTHDATE) = 12
```

MONTHNAME

MONTHNAME

►►MONTHNAME(*—expresión—*)◄◄

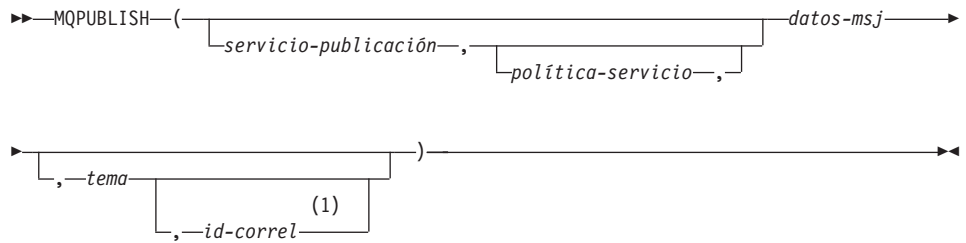
El esquema es SYSFUN.

Devuelve una serie de caracteres en mayúsculas y minúsculas mezcladas que contienen el nombre del mes (p. ej., Enero) para la parte del mes del argumento basado en el entorno nacional del momento en que se ha iniciado la base de datos.

El argumento debe ser una fecha, una indicación de la hora o una representación de serie de caracteres válida de una fecha o indicación de la hora que no sea CLOB ni LONG VARCHAR.

El resultado de la función es VARCHAR(100). El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

MQPUBLISH

**Notas:**

- 1 El *id-correl* no se puede especificar a no ser que se hayan definido anteriormente un *servicio* y una *política*.

El esquema es MQDB2.

La función MQPUBLISH publica datos en MQSeries. Esta función requiere la instalación de MQSeries Publish/Subscribe o MQSeries Integrator. Consulte www.ibm.com/software/MQSeries para obtener detalles adicionales.

La función MQPUBLISH publica los datos contenidos en *datos-msj* en el publicador de MQSeries especificado en *servicio-publicación*, utilizando la política de calidad del servicio definida por *política-servicio*. Se puede especificar un tema opcional para el mensaje y también se puede especificar un identificador de correlación de mensaje opcional definido por el usuario. La función devuelve un valor de '1' si se realiza satisfactoriamente o de '0' si no se realiza satisfactoriamente.

servicio-publicación

Serie que contiene el destino MQSeries lógico donde se debe enviar el mensaje. Si se especifica, el *servicio-publicación* debe hacer referencia a un Punto de servicio de publicación definido en el archivo de depósito AMT.XML. Un punto de servicio es un punto final lógico desde el cual se envía o se recibe un mensaje. Las definiciones de punto de servicio incluyen el nombre del Gestor de colas y de la Cola de MQSeries. Consulte el manual de MQSeries Application Messaging Interface para obtener detalles adicionales. Si no se especifica *servicio-publicación*, se utilizará DB2.DEFAULT.PUBLISHER. El tamaño máximo de *servicio-publicación* es de 48 bytes.

política-servicio

Serie que contiene la Política de servicio AMI de MQSeries que se debe utilizar en el manejo de este mensaje. Si se especifica, la *política-servicio* debe hacer referencia a una Política definida en el archivo de depósito AMT.XML. Una Política de servicio define un conjunto de opciones de

calidad de servicio que deben aplicarse a esta operación de mensajería. Estas opciones incluyen la prioridad de mensaje y la permanencia de mensaje. Consulte el manual de MQSeries Application Messaging Interface para obtener detalles adicionales. Si no se especifica *política-servicio*, se utilizará el valor por omisión DB2.DEFAULT.POLICY. El tamaño máximo de *política-servicio* es de 48 bytes.

datos-msj

Expresión de serie que contiene los datos que se deben enviar a través de MQSeries. El tamaño máximo es de 4000 bytes.

tema

Expresión de serie que contiene el tema para la publicación del mensaje. Si no se especifica ningún tema, no se asociará ninguno con el mensaje. El tamaño máximo de *tema* es de 40 bytes. Se pueden especificar múltiples temas en una serie (de una longitud máxima de 40). Cada tema debe estar separado por dos puntos. Por ejemplo, "t1:t2:el tercer tema" indica que el mensaje está asociado con los tres temas: t1, t2 y "el tercer tema".

id-correl

Expresión de serie opcional que contiene un identificador de correlación que se debe asociar con este mensaje. El *id-correl* se especifica normalmente en escenarios de petición y respuesta para asociar las peticiones con las respuestas. Si no se especifica, no se añadirá ningún id de correlación al mensaje. El tamaño máximo de *id-correl* es de 24 bytes.

Ejemplos

Ejemplo 1: En este ejemplo se publica la serie "Testing 123" en el servicio de publicación por omisión (DB2.DEFAULT.PUBLISHER) utilizando la política por omisión (DB2.DEFAULT.POLICY). No se especifica ningún identificador de correlación ni tema para el mensaje.

```
VALUES MQPUBLISH('Testing 123')
```

Ejemplo 2: En este ejemplo se publica la serie "Testing 345" en el servicio de publicación "MYPUBLISHER" bajo el tema "TESTS". Se utiliza la política por omisión y no se especifica ningún identificador de correlación.

```
VALUES MQPUBLISH('MYPUBLISHER','Testing 345','TESTS')
```

Ejemplo 3: En este ejemplo se publica la serie "Testing 678" en el servicio de publicación "MYPUBLISHER" utilizando la política "MYPOLICY" con un identificador de correlación de "TEST1". El mensaje se publica con el tema "TESTS".

```
VALUES MQPUBLISH('MYPUBLISHER','MYPOLICY','Testing 678','TESTS','TEST1')
```

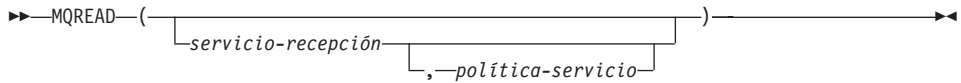
Ejemplo 4: En este ejemplo se publica la serie "Testing 901" en el servicio de publicación "MYPUBLISHER" bajo el tema "TESTS" utilizando la política por omisión (DB2.DEFAULT.POLICY) y sin identificador de correlación.

```
VALUES MQPUBLISH('Testing 901', 'TESTS')
```

Todos los ejemplos devuelve el valor '1' si la operación se realiza satisfactoriamente.

MQREAD

MQREAD



El esquema es MQDB2.

La función MQREAD devuelve un mensaje de la ubicación MQSeries especificada por *servicio-recepción*, utilizando la política de calidad de servicio definida en *política-servicio*. Al ejecutar esta operación no se elimina el mensaje de la cola asociada con *servicio-recepción*, sino que, en lugar de ello, se devuelve el mensaje situado en la cabecera de la cola. El resultado de la función es VARCHAR(4000). Si no hay mensajes disponibles para devolverse, el resultado es el valor nulo.

servicio-recepción

Serie que contiene el destino MQSeries lógico desde el que se debe recibir el mensaje. Si se especifica, el *servicio-recepción* debe hacer referencia a un Punto de servicio definido en el archivo de depósito AMT.XML. Un punto de servicio es un punto final lógico desde el que se envía o se recibe un mensaje. Las definiciones de puntos de servicio incluyen el nombre del Gestor de colas y de la Cola de MQSeries. Consulte el manual de MQSeries Application Messaging Interface para obtener detalles adicionales. Si no se especifica *servicio-recepción*, se utilizará DB2.DEFAULT.SERVICE. El tamaño máximo de *servicio-recepción* es de 48 bytes.

política-servicio

Serie que contiene la Política de servicio AMI de MQSeries utilizada en el manejo de este mensaje. Si se especifica, la *política-servicio* debe hacer referencia a una Política definida en el archivo de depósito AMT.XML. Una Política de servicio define un conjunto de opciones de calidad de servicio que deben aplicarse a esta operación de mensajería. Estas opciones incluyen la prioridad de mensaje y la permanencia de mensaje. Consulte el manual de MQSeries Application Messaging Interface para obtener detalles adicionales. Si no se especifica *política-servicio*, se utilizará el valor por omisión DB2.DEFAULT.POLICY. El tamaño máximo de *política-servicio* es de 48 bytes.

Ejemplos:

Ejemplo 1: En este ejemplo se lee el mensaje situado en la cabecera de la cola especificada por el servicio por omisión (DB2.DEFAULT.SERVICE), utilizando la política por omisión (DB2.DEFAULT.POLICY).

VALORES MQREAD()

Ejemplo 2: En este ejemplo se lee el mensaje situado en la cabecera de la cola especificada por el servicio "MYSERVICE" utilizando la política por omisión (DB2.DEFAULT.POLICY).

```
VALUES MQREAD('MYSERVICE')
```

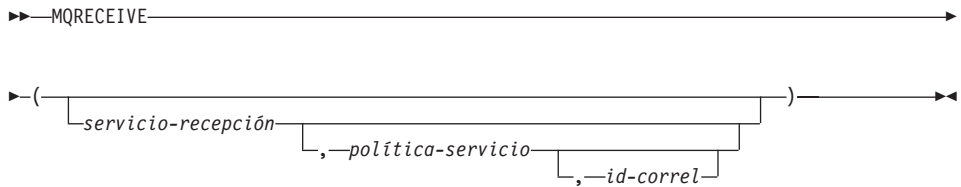
Ejemplo 3: En este ejemplo se lee el mensaje situado en la cabecera de la cola especificada por el servicio "MYSERVICE", utilizando la política "MYPOLICY".

```
VALUES MQREAD('MYSERVICE', 'MYPOLICY')
```

Todos estos ejemplos devuelven el contenido del mensaje como VARCHAR(4000), si la operación se realiza satisfactoriamente. Si no hay mensajes disponibles, el resultado es el valor nulo.

MQRECEIVE

MQRECEIVE



El esquema es MQDB2.

La función MQRECEIVE devuelve un mensaje de la ubicación MQSeries especificada por *servicio-recepción*, utilizando la política de calidad de servicio *política-servicio*. Al realizar esta operación se elimina el mensaje de la cola asociada con *servicio-recepción*. Si se especifica el *id-correl*, se devolverá el primer mensaje con un identificador de correlación que coincida. Si no se especifica *id-correl*, se devolverá el mensaje situado en la cabecera de la cola. El resultado de la función es VARCHAR(4000). Si no hay mensajes disponibles para devolverse, el resultado es el valor nulo.

servicio-recepción

Serie que contiene el destino MQSeries lógico desde el cual se recibe el mensaje. Si se especifica, el *servicio-recepción* debe hacer referencia a un Punto de servicio definido en el archivo de depósito AMT.XML. Un punto de servicio es un punto final lógico desde el cual se envía o se recibe un mensaje. Las definiciones de puntos de servicio incluyen el nombre del Gestor de colas y de la Cola de MQSeries. Consulte el manual de MQSeries Application Messaging Interface para obtener detalles adicionales. Si no se especifica *servicio-recepción*, se utiliza DB2.DEFAULT.SERVICE. El tamaño máximo de *servicio-recepción* es de 48 bytes.

política-servicio

Serie que contiene la Política de servicio AMI de MQSeries que se debe utilizar en el manejo de este mensaje. Si se especifica, la *política-servicio* debe hacer referencia a una Política definida en el archivo de depósito AMT XML⁴². Si no se especifica *política-servicio*, se utiliza el valor por omisión DB2.DEFAULT.POLICY. El tamaño máximo de *política-servicio* es de 48 bytes.

id-correl

Serie que contiene un identificación de correlación opcional que se debe

42. Una Política de servicio define un conjunto de opciones de calidad de servicio que deben aplicarse a esta operación de mensajería. Estas opciones incluyen la prioridad de mensaje y la permanencia de mensaje. Consulte el manual de MQSeries Application Messaging Interface para obtener detalles adicionales.

asociar con este mensaje. El *id-correl* se especifica normalmente en escenarios de petición y respuesta para asociar las peticiones con las respuestas. Si no se especifica, no se especificará ningún id de correlación. El tamaño máximo de *id-correl* es de 24 bytes.

Ejemplos:

Ejemplo 1: En este ejemplo se recibe el mensaje que se encuentra en la cabecera de la cola especificada por el servicio por omisión (DB2.DEFAULT.SERVICE), utilizando la política por omisión (DB2.DEFAULT.POLICY).

VALUES MQRECEIVE()

Ejemplo 2: En este ejemplo se recibe el mensaje que se encuentra en la cabecera de la cola especificada por el servicio "MYSERVICE", utilizando la política por omisión (DB2.DEFAULT.POLICY).

VALUES MQRECEIVE('MYSERVICE')

Ejemplo 3: En este ejemplo se recibe el mensaje que se encuentra en la cabecera de la cola especificada por el servicio "MYSERVICE", utilizando la política "MYPOLICY".

VALUES MQRECEIVE('MYSERVICE', 'MYPOLICY')

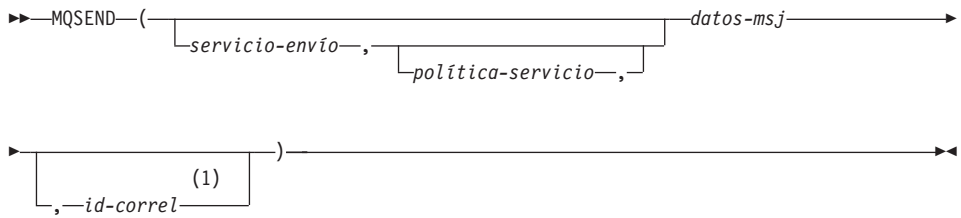
Ejemplo 4: En este ejemplo se recibe el primer mensaje con un id de correlación que coincide con '1234' de la cabecera de la cola especificada por el servicio "MYSERVICE", utilizando la política "MYPOLICY".

VALUES MQRECEIVE('MYSERVICE', 'MYPOLICY', '1234')

Todos estos ejemplos devuelven el contenido del mensaje como VARCHAR(4000), si la operación se realiza satisfactoriamente. Si no hay ningún mensaje disponible, se devolverá NULL.

MQSEND

MQSEND



Notas:

- 1 El *id-correl* no se puede especificar a no ser que se hayan definido anteriormente un *servicio* y una *política*.

El esquema es MQDB2.

La función MQSEND envía los datos contenidos en *datos-msj* a la ubicación MQSeries especificada por *servicio-envío*, utilizando la política de calidad de servicio definida por *política-servicio*. Se puede especificar un identificador de correlación de mensaje opcional definido por el usuario mediante *id-correl*. La función devuelve un valor de '1' si se realiza satisfactoriamente o de '0' si no se realiza satisfactoriamente.

datos-msj

Expresión de serie que contiene los datos que se deben enviar a través de MQSeries. El tamaño máximo es de 4000 bytes.

servicio-envío

Serie que contiene el destino MQSeries lógico donde se debe enviar el mensaje. Si se especifica, el *servicio-envío* hace referencia a un punto de servicio definido en el archivo de depósito AMT.XML. Un punto de servicio es un punto final lógico desde el cual se puede enviar o recibir un mensaje. Las definiciones de punto de servicio incluyen el nombre del Gestor de colas y de la Cola de MQSeries. Consulte el manual de MQSeries Application Messaging Interface para obtener detalles adicionales. Si no se especifica *servicio-envío*, se utiliza el valor de DB2.DEFAULT.SERVICE. El tamaño máximo de *servicio-envío* es de 48 bytes.

política-servicio

Serie que contiene la Política de servicio AMI de MQSeries utilizada en el manejo de este mensaje. Si se especifica, la *política-servicio* debe hacer referencia a una política de servicio definida en el archivo de depósito AMT XML. Una Política de servicio define un conjunto de opciones de calidad de servicio que deben aplicarse a esta operación de mensajería. Estas opciones incluyen la prioridad de mensaje y la permanencia de

mensaje. Consulte el manual de MQSeries Application Messaging Interface para obtener detalles adicionales. Si no se especifica *política-servicio*, se utilizará un valor por omisión de DB2.DEFAULT.POLICY. El tamaño máximo de *política-servicio* es de 48 bytes.

id-correl

Serie opcional que contiene un identificador de correlación asociado con este mensaje. El *id-correl* se especifica normalmente en escenarios de petición y respuesta para asociar las peticiones con las respuestas. Si no se especifica, no se enviará ningún id de correlación. El tamaño máximo de *id-correl* es de 24 bytes.

Ejemplos:

Ejemplo 1: En este ejemplo se envía la serie "Testing 123" al servicio por omisión (DB2.DEFAULT.SERVICE), utilizando la política por omisión (DB2.DEFAULT.POLICY), sin identificador de correlación.

```
VALUES MQSEND('Testing 123')
```

Ejemplo 2: En este ejemplo se envía la serie "Testing 345" al servicio "MYSERVICE", utilizando la política "MYPOLICY", sin identificador de correlación.

```
VALUES MQSEND('MYSERVICE', 'MYPOLICY', 'Testing 345')
```

Ejemplo 3: En este ejemplo se envía la serie "Testing 678" al servicio "MYSERVICE", utilizando la política "MYPOLICY", con el identificador de correlación "TEST3".

```
VALUES MQSEND('MYSERVICE', 'MYPOLICY', 'Testing 678', 'TEST3')
```

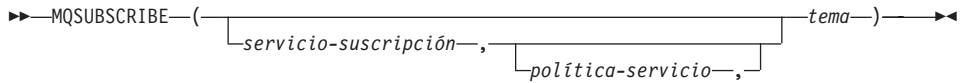
Ejemplo 4: En este ejemplo se envía la serie "Testing 901" al servicio "MYSERVICE", utilizando la política por omisión (DB2.DEFAULT.POLICY) y ningún identificador de correlación.

```
VALUES MQSEND('MYSERVICE', 'Testing 901')
```

Todos los ejemplos devuelven un valor escalar de '1' si la operación se realiza satisfactoriamente.

MQSUBSCRIBE

MQSUBSCRIBE



El esquema es MQDB2.

La función MQSUBSCRIBE se utiliza para registrar interés en los mensajes MQSeries publicados sobre un tema especificado. El *servicio-suscripción* especifica un destino lógico para los mensajes que coinciden con el tema especificado. Los mensajes que coinciden con el *tema* se colocarán en la cola definida por *servicio-suscripción* y se podrán leer o recibir mediante una llamada subsiguiente a MQREAD, MQRECEIVE, MQREADALL o MQRECEIVEALL. Esta función requiere que se instale y se configure un sistema de publicación y suscripción basado en MQSeries, por ejemplo MQSeries Integrator o MQSeries Publish/Subscribe. Consulte www.ibm.com/software/MQSeries para obtener detalles adicionales.

La función devuelve un valor de '1' si se realiza satisfactoriamente o de '0' si no se realiza satisfactoriamente. La ejecución satisfactoria de esta función hará que el servidor de publicación y suscripción reenvíe los mensajes que coinciden con el tema al punto de servicio definido por *servicio-suscripción*.

servicio-suscripción

Serie que contiene el punto de suscripción MQSeries lógico al que se enviarán los mensajes que coinciden con *tema*. Si se especifica, el *servicio-suscripción* debe hacer referencia a un Punto de servicio de suscriptores definido en el archivo de depósito AMT.XML. Las definiciones de puntos de servicio incluyen el nombre del Gestor de colas y de la Cola de MQSeries. Consulte el manual de MQSeries Application Messaging Interface para obtener detalles adicionales. Si no se especifica *servicio-suscripción*, en su lugar se utilizará DB2.DEFAULT.SUBSCRIBER. El tamaño máximo de *servicio-suscripción* es de 48 bytes.

política-servicio

Serie que contiene la Política de servicio AMI de MQSeries que se debe utilizar en el manejo del mensaje. Si se especifica, la *política-servicio* debe hacer referencia a una Política definida en el archivo de depósito AMT.XML. Una Política de servicio define un conjunto de opciones de calidad de servicio que se deben aplicar a esta operación de mensajería. Estas opciones incluyen la prioridad de mensaje y la permanencia de mensaje. Consulte el manual de MQSeries Application Messaging Interface para obtener detalles adicionales. Si no se especifica *política-servicio*, en su lugar se utilizará DB2.DEFAULT.POLICY. El tamaño máximo de *política-servicio* es de 48 bytes.

tema

Serie que define los tipos de mensajes que se deben recibir. Esta suscripción sólo recibirá los mensajes publicados con los temas especificados. Pueden coexistir múltiples suscripciones. El tamaño máximo es de 40 bytes. Se pueden especificar múltiples temas en una serie (de una longitud máxima de 40). Cada tema debe estar separado por dos puntos. Por ejemplo, "t1:t2:el tercer tema" indica que el mensaje está asociado con los tres temas: t1, t2 y "el tercer tema".

Ejemplos:

Ejemplo 1: En este ejemplo se registra un interés en los mensajes que contienen el tema "Tiempo". El servicio-suscripción por omisión (DB2.DEFAULT.SUBSCRIBER) se registra como suscriptor y la política-servicio por omisión (DB2.DEFAULT.POLICY) especifica la calidad de servicio.

VALUES MQSUBSCRIBE('Tiempo')

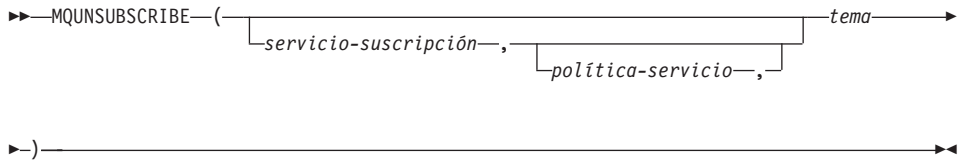
Ejemplo 2: Este ejemplo muestra un interés de registro de suscriptor en los mensajes que contienen "Stocks". El suscriptor se registra como "PORTFOLIO-UPDATES" con la política "BASIC-POLICY".

VALUES MQSUBSCRIBE('PORTFOLIO-UPDATES','BASIC-POLICY','Stocks')

Todos los ejemplos devuelven un valor escalar de '1' si la operación se realiza satisfactoriamente.

MQUNSUBSCRIBE

MQUNSUBSCRIBE



El esquema es MQDB2.

La función MQUNSUBSCRIBE se utiliza para eliminar el registro de una suscripción de mensajes existente. El *servicio-suscripción*, la *política-servicio* y el *tema* se utilizan para identificar qué suscripción se cancela. Esta función requiere que se instale y se configure un sistema de publicación y suscripción basado en MQSeries, por ejemplo MQSeries Integrator o MQSeries Publish/Subscribe. Consulte www.ibm.com/software/MQSeries para obtener detalles adicionales.

La función devuelve un valor de '1' si se realiza satisfactoriamente o de '0' si no se realiza satisfactoriamente. El resultado de la ejecución satisfactoria de esta función consiste en que el servidor de publicación y suscripción elimina la suscripción definida por los parámetros proporcionados. Los mensajes con el *tema* especificado no se enviarán ya al destino lógico definido por *servicio-suscripción*.

servicio-suscripción

Si se especifica, el *servicio-suscripción* debe hacer referencia a un Punto de servicio de suscriptores definido en el archivo de depósito AMT.XML. Las definiciones de punto de servicio incluyen el nombre del Gestor de colas y de la Cola de MQSeries. Consulte el manual de MQSeries Application Messaging Interface para obtener detalles adicionales. Si no se especifica *servicio-suscripción*, se utilizará el valor DB2.DEFAULT.SUBSCRIBER. El tamaño máximo de *servicio-suscripción* es de 48 bytes.

política-servicio

Si se especifica, la *política-servicio* debe hacer referencia a una Política definida en el archivo de depósito AMT.XML. Una Política de servicio define un conjunto de opciones de calidad de servicio que se deben aplicar a esta operación de mensajería. Consulte el manual de MQSeries Application Messaging Interface para obtener detalles adicionales. Si no se especifica *política-servicio*, se utilizará el valor por omisión DB2.DEFAULT.POLICY. El tamaño máximo de *política-servicio* es de 48 bytes.

tema

Serie que especifica el tema de los mensajes que no se deben recibir. El

tamaño máximo de *tema* es de 40 bytes. Se pueden especificar múltiples temas en una serie (de una longitud máxima de 40). Cada tema debe estar separado por dos puntos. Por ejemplo, "t1:t2:el tercer tema" indica que el mensaje está asociado con los tres temas: t1, t2 y "el tercer tema".

Ejemplos:

Ejemplo 1: En este ejemplo se cancela un interés en los mensajes que contienen el tema "Tiempo". El servicio-suscripción por omisión (DB2.DEFAULT.SUBSCRIBER) se registra como servicio que cancela la suscripción y la política-servicio por omisión (DB2.DEFAULT.POLICY) especifica la calidad de servicio.

VALUES MQUNSUBSCRIBE('Tiempo')

Ejemplo 2: Este ejemplo muestra un suscriptor que cancela un interés en los mensajes que contienen "Stocks". El suscriptor se registra como "PORTFOLIO-UPDATES" con la política "BASIC-POLICY".

VALUES MQUNSUBSCRIBE('PORTFOLIO-UPDATES', 'BASIC-POLICY', 'Stocks')

Estos ejemplos devuelven un valor escalar de '1' si la operación se realiza satisfactoriamente y un valor escalar de '0' si la operación no se realiza satisfactoriamente.

MULTIPLY_ALT

MULTIPLY_ALT

►►MULTIPLY_ALT(*—expresión_numérica_exacta—*,*—expresión_numérica_exacta—*)◄◄

El esquema es SYSIBM.

La función escalar MULTIPLY_ALT devuelve el producto de los dos argumentos como un valor decimal. Se proporciona como alternativa al operador de multiplicación, especialmente cuando la suma de las precisiones de los argumentos excede de 31.

Los argumentos pueden ser cualquier tipo de datos numéricos exactos incorporado (DECIMAL, BIGINT, INTEGER o SMALLINT).

El resultado de la función es un DECIMAL. La precisión y la escala del resultado se determinan del modo siguiente, utilizando los símbolos p y s para indicar la precisión y la escala del primer argumento y los símbolos p' y s' para indicar la precisión y la escala del segundo argumento.

La precisión es $\text{MIN}(31, p + p')$

La escala es:

- 0 si la escala de ambos argumentos es 0
- $\text{MIN}(31, s + s')$ si $p + p'$ es inferior o igual a 31
- $\text{MAX}(\text{MIN}(3, s + s'), 31 - (p - s + p' - s'))$ si $p + p'$ es superior a 31.

el resultado puede ser nulo si al menos un argumento puede ser nulo o si la base de datos se configura con DFT_SQLMATHWARN establecido en YES; el resultado es el valor nulo si uno de los argumentos es nulo.

Es mejor elegir la función MULTIPLY_ALT que el operador de multiplicación cuando se realizan operaciones aritméticas decimales donde se necesita como mínimo una escala de 3 y la suma de las precisiones excede de 31. En estos casos, se realiza el cálculo interno a fin de evitar desbordamientos. Entonces se asigna el resultado final al tipo de datos de resultado, utilizando el truncamiento donde sea necesario para coincidir con la escala. Tenga en cuenta que el desbordamiento del resultado final sigue siendo posible cuando la escala es 3.

A continuación se proporciona un ejemplo en el que se comparan los tipos de resultado utilizando MULTIPLY_ALT y el operador de multiplicación.

Tipo de argumento 1	Tipo de argumento 2	Resultado utilizando MULTIPLY_ALT	Resultado utilizando el operador de multiplicación
DECIMAL(31,3)	DECIMAL(15,8)	DECIMAL(31,3)	DECIMAL(31,11)
DECIMAL(26,23)	DECIMAL(10,1)	DECIMAL(31,19)	DECIMAL(31,24)
DECIMAL(18,17)	DECIMAL(20,19)	DECIMAL(31,29)	DECIMAL(31,31)
DECIMAL(16,3)	DECIMAL(17,8)	DECIMAL(31,9)	DECIMAL(31,11)
DECIMAL(26,5)	DECIMAL(11,0)	DECIMAL(31,3)	DECIMAL(31,5)
DECIMAL(21,1)	DECIMAL(15,1)	DECIMAL(31,2)	DECIMAL(31,2)

Ejemplo:

Multiplique dos valores donde el tipo de datos del primer argumento es DECIMAL(26,3) y el tipo de datos del segundo argumento es DECIMAL(9,8). El tipo de datos del resultado es DECIMAL(31,7).

```
values multiply_alt(98765432109876543210987.654,5.43210987)
1 -----
536504678578875294857887.5277415
```

Observe que el producto completo de estos dos números es 536504678578875294857887.52774154498, pero los 4 últimos dígitos están truncados para coincidir con la escala del tipo de datos de resultado. Si se utiliza el operador de multiplicación con los mismos valores se producirá un desbordamiento aritmético, dado que el tipo de datos de resultado es DECIMAL(31,11) y el valor de resultado tiene 24 dígitos a la izquierda del decimal, pero el tipo de datos de resultado sólo soporta 20 dígitos.

NODENUMBER

NODENUMBER

►—NODENUMBER—(*—nombre-columna—*)—◄

El esquema es SYSIBM.

La función NODENUMBER devuelve el número de partición de la fila. Por ejemplo, si se utiliza en una cláusula SELECT, devuelve el número de partición para cada fila de la tabla que se ha utilizado para formar el resultado de la sentencia SELECT.

El número de partición devuelto en las tablas y variables de transición procede de los valores de transición actuales de las columnas de claves de particionamiento. Por ejemplo, en un desencadenante BEFORE INSERT, la función devolverá el número de partición proyectado teniendo en cuenta los valores actuales de las nuevas variables de transición. No obstante, un desencadenante BEFORE INSERT subsiguiente puede modificar los valores de las columnas de claves de particionamiento. De este modo, el número de partición final de la fila cuando se inserte en la base de datos puede ser diferente del valor proyectado.

El argumento debe ser el nombre calificado o no calificado de una columna de una tabla. La columna puede tener cualquier tipo de datos.⁴³ Si *nombre-columna* hace referencia a una columna de una vista, la expresión de la vista para la columna debe hacer referencia a una columna de la tabla base principal y la vista debe ser suprimible. Una expresión de tabla anidada o común sigue las mismas reglas que una vista. Consulte el apartado “Notas” en la página 966 para ver la definición de una vista suprimible.

La fila específica (y tabla) para la que se devuelve el número de partición mediante la función NODENUMBER se determina por el contexto de la sentencia de SQL que utiliza la función.

El tipo de datos del resultado es INTEGER y nunca es nulo. Puesto que se devuelve información a nivel de fila, los resultados son los mismos, sin tener en cuenta las columnas que se especifican para la tabla. Si no hay ningún archivo `db2nodes.cfg`, el resultado es 0.

43. Esta función no puede utilizarse como una función fuente cuando se crea una función definida por el usuario. Como acepta cualquier tipo de datos como argumento, no es necesario crear firmas adicionales para dar soporte a los tipos diferenciados definidos por el usuario.

La función NODENUMBER no puede utilizarse en tablas duplicadas, dentro de restricciones de comprobación ni en la definición de columnas generadas (SQLSTATE 42881).

Ejemplos:

- Cuente el número de filas en las que la fila para un EMPLOYEE está en una partición diferente de la descripción del departamento del empleado en DEPARTMENT.

```
SELECT COUNT(*) FROM DEPARTMENT D, EMPLOYEE E  
WHERE D.DEPTNO=E.WORKDEPT  
AND NODENUMBER(E.LASTNAME) <> NODENUMBER(D.DEPTNO)
```

- Una las tablas EMPLOYEE y DEPARTMENT donde las filas de las dos tablas estén en la misma partición.

```
SELECT * FROM DEPARTMENT D, EMPLOYEE E  
WHERE NODENUMBER(E.LASTNAME) = NODENUMBER(D.DEPTNO)
```

- Anote cronológicamente el número de empleado y el número de partición proyectado de la nueva fila en una tabla denominada EMPINSERTLOG1 para cualquier inserción de empleados creando un desencadenante BEFORE en la tabla EMPLOYEE.

```
CREATE TRIGGER EMPINSLOGTRIG1  
BEFORE INSERT ON EMPLOYEE  
REFERENCING NEW AS NEWTABLE  
FOR EACH MODE ROW MODE DB2SQL  
INSERT INTO EMPINSERTLOG1  
VALUES(NEWTABLE.EMPNO, NODENUMBER(NEWTABLE.EMPNO))
```

NULLIF

NULLIF

►—NULLIF—(—*expresión*—,—*expresión*—)——►

El esquema es SYSIBM.

La función NULLIF devuelve un valor nulo si los argumentos son iguales, de lo contrario, devuelve el valor del primer argumento.

Los argumentos deben poderse comparar (consulte el apartado “Asignaciones y comparaciones” en la página 105). Pueden ser de un tipo de datos incorporado (que no sea una serie larga ni DATALINK) o de un tipo de datos diferenciado (que no esté basado en una serie larga ni en DATALINK).⁴⁴ Los atributos del resultado son los atributos del primer argumento.

El resultado de utilizar NULLIF(e1,e2) es igual a utilizar la expresión

```
CASE WHEN e1=e2 THEN NULL ELSE e1 END
```

Observe que cuando e1=e2 se evalúa como desconocido (porque uno o los dos argumentos son NULL), las expresiones CASE lo consideran como no verdadero. Por lo tanto, en esta situación, NULLIF devuelve el valor del primer argumento.

Ejemplo:

- Suponga que las variables PROFIT, CASH y LOSSES tienen tipos de datos DECIMAL con los valores 4500.00, 500.00 y 5000.00 respectivamente:

```
NULLIF (:PROFIT + :CASH , :LOSSES )
```

Devuelve un valor nulo.

44. Esta función no puede utilizarse como una función fuente cuando se crea una función definida por el usuario. Como acepta cualquier tipo de datos compatible como argumento, no es necesario crear firmas adicionales para dar soporte a los tipos de datos diferenciados definidos por el usuario.

PARTITION

►►—PARTITION—(—*nombre-columna*—)——►►

El esquema es SYSIBM.

La función PARTITION devuelve el índice de mapa de particionamiento de la fila obtenido mediante la aplicación de la función de partición en el valor de clave de particionamiento de la fila. Por ejemplo, si se utiliza en una cláusula SELECT, devuelve el índice de mapa de particionamiento de cada fila de la tabla que se ha utilizado para formar el resultado de la sentencia SELECT.

El índice de mapa de particionamiento devuelto en las tablas y variables de transición procede de los valores de transición actuales de las columnas de claves de particionamiento. Por ejemplo, en un desencadenante BEFORE INSERT, la función devolverá el índice de mapa de particionamiento proyectado teniendo en cuenta los valores actuales de las nuevas variables de transición. No obstante, un desencadenante BEFORE INSERT subsiguiente puede modificar los valores de las columnas de claves de particionamiento. De este modo, el índice de mapa de particionamiento final de la fila cuando se inserte en la base de datos puede ser diferente del valor proyectado.

El argumento debe ser el nombre calificado o no calificado de una columna de una tabla. La columna puede tener cualquier tipo de datos.⁴⁵ Si *nombre-columna* hace referencia a una columna de una vista, la expresión de la vista para la columna debe hacer referencia a una columna de la tabla base principal y la vista debe ser suprimible. Una expresión de tabla anidada o común sigue las mismas reglas que una vista. Consulte el apartado “Notas” en la página 966 para ver la definición de una vista suprimible.

La fila específica (y la tabla) para la que la función PARTITION devuelve el índice de mapa de particionamiento se determina por el contexto de la sentencia de SQL que utiliza la función.

El tipo de datos del resultado es INTEGER en el rango de 0 a 4095. Para una tabla sin clave de particionamiento, el resultado siempre es 0. No se devuelve nunca un valor nulo. Puesto que se devuelve información a nivel de fila, los resultados son los mismos, sin tener en cuenta las columnas que se especifican para la tabla.

45. Esta función no puede utilizarse como una función fuente cuando se crea una función definida por el usuario. Como acepta cualquier tipo de datos como argumento, no es necesario crear firmas adicionales para dar soporte a los tipos diferenciados definidos por el usuario.

PARTITION

La función PARTITION no puede utilizarse en tablas duplicadas, dentro de restricciones de comprobación ni en la definición de columnas generadas (SQLSTATE 42881).

Ejemplo:

- Liste los números de empleado (EMPNO) de la tabla EMPLOYEE para todas las filas con un índice de mapa de particionamiento de 100.

```
SELECT EMPNO FROM EMPLOYEE  
WHERE PARTITION(PHONENO) = 100
```

- Anote el número de empleado y el índice de mapa de particionamiento previsto de la nueva fila en una tabla denominada EMPINSERTLOG2 para cualquier inserción de empleados creando un desencadenante BEFORE en la tabla EMPLOYEE.

```
CREATE TRIGGER EMPINSLOGTRIG2  
BEFORE INSERT ON EMPLOYEE  
REFERENCING NEW AS NEWTABLE  
FOR EACH MODE ROW MODE DB2SQL  
INSERT INTO EMPINSERTLOG2  
VALUES(NEWTABLE.EMPNO, PARTITION(NEWTABLE.EMPNO))
```


POSSTR

►►—POSSTR—(—*serie-fuente*—,—*serie-búsqueda*—)——————►►

El esquema es SYSIBM.

La función POSSTR devuelve la posición inicial de la primera ocurrencia de una serie (llamada la *serie-búsqueda*) en otra serie (llamada la *serie-fuente*). Los números para la posición de *serie-búsqueda* empiezan en 1 (no en 0).

El resultado de la función es un entero grande. Si alguno de los argumentos puede ser nulo, el resultado puede ser nulo; si alguno de los argumentos es nulo, el resultado es el valor nulo.

serie-fuente

Una expresión que especifica la serie fuente en la que se ha de llevar a cabo la búsqueda.

La expresión puede especificarse mediante:

- una constante
- un registro especial
- una variable del lenguaje principal (como por ejemplo, una variable localizadora o una variable de referencia a archivos)
- una función escalar
- un localizador de gran objeto
- un nombre de columna
- una expresión que concatene cualquiera de los elementos anteriores

serie-búsqueda

Una expresión que especifica la serie que se ha de buscar.

La expresión puede especificarse mediante:

- una constante
- un registro especial
- una variable del lenguaje principal
- una función escalar cuyos operandos sean cualquiera de los elementos anteriores
- una expresión que concatene cualquiera de los elementos anteriores

teniendo en cuenta las siguientes restricciones:

- Ningún elemento de la expresión puede ser de tipo LONG VARCHAR, CLOB, LONG VARGRAPHIC o DBCLOB. Además, no puede ser una variable de referencia a archivos BLOB.

- La longitud real de *serie-búsqueda* no puede tener más de 4.000 bytes.

Tenga en cuenta que estas reglas son iguales que las de la *expresión-patrón* descrita en el apartado “Predicado LIKE” en la página 223.

Tanto la *serie-búsqueda* como la *serie-fuente* tienen cero o más posiciones continuas. Si las series son series de caracteres o binarias, una posición es un byte. Si las series son series gráficas, una posición es un carácter gráfico (DBCS).

La función POSSTR acepta las series de datos mixtos. Sin embargo, POSSTR opera sobre la base de un número de bytes estricto, ajeno a los cambios entre caracteres de un solo byte o de múltiples bytes.

Se aplican las siguientes reglas:

- Los tipos de datos de la *serie-fuente* y la *serie-búsqueda* deben ser compatibles, de lo contrario se genera un error (SQLSTATE 42884).
 - Si *serie-fuente* es una serie de caracteres, entonces *serie-búsqueda* ha de ser una serie de caracteres, pero no una CLOB ni LONG VARCHAR, con una longitud real de 32 672 bytes o menos.
 - Si *serie-fuente* es una serie gráfica, entonces *serie-búsqueda* ha de ser una serie gráfica, pero no una DBCLOB ni LONG VARGRAPHIC, con una longitud real de 16 336 caracteres de doble byte o menos.
 - Si *serie-fuente* es una serie binaria, entonces *serie-búsqueda* ha de ser una serie binaria con una longitud real de 32 672 bytes o menos.
- Si la *serie-búsqueda* tiene una longitud de cero, el resultado que devuelve la función es 1.
- De lo contrario:
 - Si la *serie-fuente* tiene una longitud de cero, el resultado que devuelve la función es cero.
 - De lo contrario:
 - Si el valor de la *serie-búsqueda* es igual a una subserie de longitud idéntica de posiciones continuas del valor de la *serie-fuente*, el resultado devuelto por la función es la posición inicial de la primera de dicha subserie en el valor *serie-fuente*.
 - De lo contrario, el resultado que devuelve la función es 0.

Ejemplo

- Seleccione las columnas RECEIVED y SUBJECT así como la posición inicial de las palabras 'GOOD BEER' de la columna NOTE_TEXT para todas las entradas de la tabla IN_TRAY que contienen estas palabras.

```
SELECT RECEIVED, SUBJECT, POSSTR(NOTE_TEXT, 'GOOD BEER')  
FROM IN_TRAY  
WHERE POSSTR(NOTE_TEXT, 'GOOD BEER') <> 0
```

POWER

POWER

►►—POWER—(—*expresión1*—,—*expresión2*—)——————►►

El esquema es SYSFUN.

Devuelve el valor de *expresión1* elevado a la potencia de *expresión2*.

Los argumentos pueden ser de cualquier tipo de datos numérico incorporado. Los argumentos DECIMAL y REAL se convierten a un número de coma flotante de precisión doble.

El resultado de la función es:

- INTEGER si ambos argumentos son INTEGER o SMALLINT
- BIGINT si un argumento es BIGINT y el otro argumento es BIGINT, INTEGER o SMALLINT
- de lo contrario, DOUBLE.

El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

QUARTER

►►—QUARTER—(—*expresión*—)—————►►

El esquema es SYSFUN.

Devuelve un valor entero en el rango de 1 a 4 que representa el trimestre del año para la fecha especificada en el argumento.

El argumento debe ser una fecha, una indicación de la hora o una representación de serie de caracteres válida de una fecha o indicación de la hora que no sea CLOB ni LONG VARCHAR.

El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

RADIANS

RADIANS

►►RADIANS—(*—expresión—*)—►►

El esquema es SYSFUN.

Devuelve el número de radianes convertidos del argumento que se expresa en grados.

El argumento puede ser de cualquier tipo de datos numérico incorporado. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

RAISE_ERROR

►►—RAISE_ERROR—(—*sqlstate*—,—*serie-diagnóstico*—)—————►►

El esquema es SYSIBM.

La función RAISE_ERROR hace que la sentencia que incluye la función devuelva un error especificando el SQLSTATE, SQLCODE -438 y la *serie-diagnóstico*. La función RAISE_ERROR siempre devuelve NULL con un tipo de datos no definido.

sqlstate

Una serie de caracteres que contiene 5 caracteres exactamente. Debe ser de tipo CHAR definido con una longitud de 5 o un tipo VARCHAR definido con una longitud de 5 o mayor. El valor de *sqlstate* debe seguir las reglas de los SQLSTATE definidos por la aplicación, de la siguiente manera:

- Cada carácter debe pertenecer al conjunto de dígitos (del '0' al '9') o de letras mayúsculas no acentuadas (de la 'A' a la 'Z')
- La clase SQLSTATE (primeros dos caracteres) no puede ser '00', '01' ni '02', pues no son clases de error.
- Si la clase SQLSTATE (primeros dos caracteres) empieza por un carácter de los rangos 0-6 o A-H, la subclase (tres últimos caracteres) debe empezar por una letra del rango I-Z
- Si la clase SQLSTATE (primeros dos caracteres) empieza por un carácter de los rangos 7-9 o I-Z, la subclase (tres últimos caracteres) puede ser un valor cualquiera de 0-9 o A-Z.

Si SQLSTATE no se ajusta a estas reglas se produce un error (SQLSTATE 428B3).

serie-diagnóstico

Una expresión de tipo CHAR o VARCHAR que devuelve una serie de caracteres de un máximo de 70 bytes que describe la condición de error. Si la serie tiene más de 70 bytes de longitud, se truncará.

Para utilizar esta función en un contexto en el que las Reglas para los tipos de datos del resultado no se aplican (por ejemplo, solo en una lista de selección), se debe utilizar una especificación de conversión (cast) para dar un tipo de datos al valor nulo devuelto. La función RAISE_ERROR será más útil en una expresión CASE.

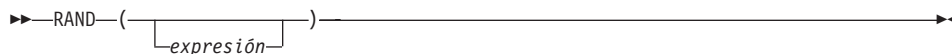
Ejemplo:

RAISE_ERROR

Liste los números de empleados y los niveles de formación como, por ejemplo, Postgraduado, Graduado y Diplomado. Si el nivel de formación es mayor que 20, genere un error.

```
SELECT EMPNO,  
       CASE WHEN EDUCLVL < 16 THEN 'Diplomado'  
            WHEN EDUCLVL < 18 THEN 'Graduado'  
            WHEN EDUCLVL < 21 THEN 'Postgraduado'  
            ELSE RAISE_ERROR('70001',  
                             'EDUCLVL tiene un valor mayor que 20')  
       END  
FROM EMPLOYEE
```


RAND



El esquema es SYSFUN.

Devuelve un valor aleatorio de coma flotante, comprendido entre 0 y 1, utilizando el argumento como valor generador opcional. La función se define como no determinista.

No es necesario ningún argumento, pero si se especifica puede ser INTEGER o SMALLINT.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

REAL

REAL

►►—REAL—(—expresión-numérica—)—————►►

El esquema es SYSIBM.

La función REAL devuelve la representación de coma flotante de precisión simple correspondiente a un número.

El argumento es una expresión que devuelve un valor de cualquier tipo de datos numérico incorporado.

El resultado de la función es un número de coma flotante de precisión simple. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

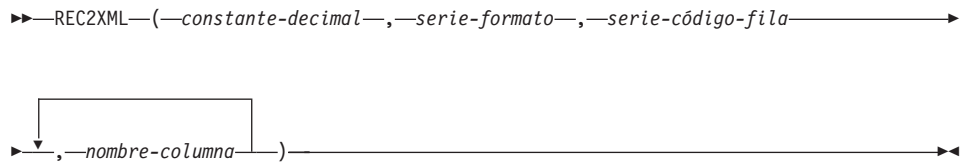
El resultado es el mismo número que sería si el argumento se hubiese asignado a una columna o variable de coma flotante de precisión simple.

Ejemplo:

Utilizando la tabla EMPLOYEE, busque la proporción de salario y comisiones para los empleados cuya comisión no sea cero. Las columnas implicadas (SALARY y COMM) tienen tipos de datos DECIMAL. El resultado se desea en coma flotante de precisión simple. Por consiguiente, se aplica REAL a SALARY para que la división se lleve a cabo en coma flotante (en realidad en precisión doble) y después se aplica REAL a la expresión completa para devolver el resultado en coma flotante de precisión simple.

```
SELECT EMPNO, REAL(REAL(SALARY)/COMM)
      FROM EMPLOYEE
WHERE COMM > 0
```

REC2XML



El esquema es SYSIBM.

La función REC2XML devuelve una serie que está formateada con códigos XML y que contiene nombres de columna y datos de columna.

constante-decimal

Factor de expansión para sustituir caracteres de datos de columna. El valor decimal debe ser mayor que 0.0 y menor que o igual a 6.0. (SQLSTATE 42820).

El valor *constante-decimal* se utiliza para calcular la longitud de resultado de la función. Para cada columna con un tipo de datos de tipo carácter, el atributo de longitud de la columna se multiplica por este factor de expansión antes de que se sume a la longitud de resultado.

Para especificar que no haya expansión, utilice un valor de 1.0. Si se especifica un valor menor que 1.0, se reducirá la longitud de resultado calculada. Si la longitud real de la serie de resultado es mayor que la longitud de resultado calculada de la función, se producirá un error (SQLSTATE 22001).

serie-formato

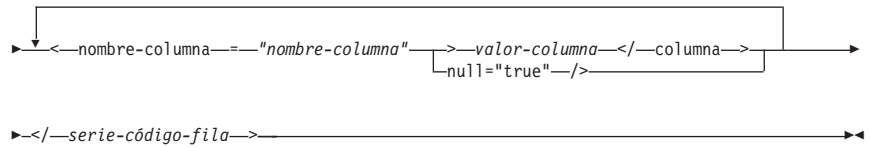
Constante de serie que especifica qué formato debe utilizar la función durante la ejecución.

La *serie-formato* es sensible a las mayúsculas y minúsculas, de modo que los valores siguientes deben especificarse en mayúsculas para que se reconozcan.

COLATTVAL o COLATTVAL_XML

Estos formatos devuelven una serie con columnas como valores de atributo.





Los nombres de columna pueden ser valores de atributo XML válidos o pueden no serlo. Para los nombres de columna que no son valores de atributo XML válidos, se realiza la sustitución de caracteres en el nombre de columna antes de incluirlo en la serie de resultado.

Los valores de columna pueden ser nombres de elemento XML válidos o pueden no serlo. Si se especifica la *serie-formato* COLATTVAL, para los nombres de columna que no son valores de elemento XML válidos, se realiza la sustitución de caracteres en el valor de columna antes de incluirlo en la serie de resultado. Si se especifica la *serie-formato* COLATTVAL_XML, no se realiza la sustitución de caracteres en los valores de columna (aunque la sustitución de caracteres se realiza de todas formas en los nombres de columna).

serie-código-fila

Constante de serie que especifica el código utilizado para cada fila. Si se especifica una serie vacía, se supone un valor de 'row'.

Si se especifica una serie de uno o más caracteres en blanco, no aparecerá ninguna *serie-código-fila* inicial o *serie-código-fila* final (incluidos los delimitadores de signo mayor que y menor que) en la serie de resultado.

nombre-columna

Nombre calificado o no calificado de una columna de tabla. La columna debe tener uno de los tipos de datos siguientes (SQLSTATE 42815):

- numérico (SMALLINT, INTEGER, BIGINT, DECIMAL, REAL, DOUBLE)
- serie de caracteres (CHAR, VARCHAR)⁴⁶
- fecha y hora (DATE, TIME, TIMESTAMP)
- un tipo definido por el usuario basado en uno de los tipos anteriores

No se puede especificar más de una vez el mismo nombre de columna (SQLSTATE 42734).

El resultado de la función es VARCHAR. La longitud máxima es de 32.672 bytes (SQLSTATE 54006).

46. No se permite una serie de caracteres con un subtipo de BIT DATA.

Examine la invocación siguiente:

REC2XML (dc, fs, rt, c₁, c₂, ..., c_n)

Si el valor de "fs" es "COLATTVAL" o "COLATTVAL_XML", el resultado será igual que esta expresión:

```
'<' CONCAT rt CONCAT '>' CONCAT y1 CONCAT y2
CONCAT ... CONCAT yn CONCAT '</' CONCAT rt CONCAT '>'
```

donde y_n es equivalente a:

```
'<column name="' CONCAT xvcn CONCAT vn
```

y vn es equivalente a:

```
'">' CONCAT rn CONCAT '</column>'
```

si la columna no es nula y

```
'" null="true"/>'
```

si el valor de column es nulo.

xvc_n es equivalente a una representación de serie del nombre de columna de c_n, donde cualquier carácter que aparezca en la Tabla 17 en la página 408 se sustituye por la representación correspondiente. Esto asegura que la serie resultante sea un símbolo de valor de elemento o atributo XML válido.

r_n es equivalente a una representación de serie como se indica en la Tabla 16

Tabla 16. Resultado de serie de valores de columna

Tipo de datos de c _n	r _n
CHAR, VARCHAR	El valor es una serie. Si la <i>serie-formato</i> no termina con los caracteres "_XML", cada carácter de c _n se sustituye por la representación de sustitución correspondiente de la Tabla 17 en la página 408, como se indica. El atributo de longitud es: dc * el atributo de longitud de c _n .
SMALLINT, INTEGER, BIGINT, DECIMAL, NUMERIC, REAL, DOUBLE	El valor es LTRIM(RTRIM(CHAR(c _n))). El atributo de longitud es la longitud de resultado de CHAR(c _n). El carácter decimal es siempre el carácter de punto ('.').
DATE	El valor es CHAR(c _n ,ISO). El atributo de longitud es la longitud de resultado de CHAR(c _n ,ISO).

Tabla 16. Resultado de serie de valores de columna (continuación)

Tipo de datos de c_n	r_n
TIME	El valor es CHAR(c_n ,JIS). El atributo de longitud es la longitud de resultado de CHAR(c_n ,JIS)
TIMESTAMP	El valor es CHAR(c_n). El atributo de longitud es la longitud de resultado de CHAR(c_n).

Sustitución de caracteres:

En función del valor especificado para la *serie-formato*, se sustituirán determinados caracteres en los nombres de columna y en los valores de columna para asegurar que los nombres de columna formen valores de atributo XML válidos y que los valores de columna formen valores de elemento XML válidos.

Tabla 17. Sustituciones de caracteres para valores de atributo y valores de elemento XML

Carácter	Sustitución
<	<
>	>
"	"
&	&
'	'

Ejemplos:

Nota: REC2XML no inserta caracteres de nueva línea en la salida. Se formatea toda la salida de ejemplo para que se pueda leer.

- Utilizando la tabla DEPARTMENT de la base de datos de ejemplo, formatee la fila de tabla de departamento, excepto las columnas DEPTNAME y LOCATION, para el departamento 'D01' en una serie XML. Dado que los datos no contienen ninguno de los caracteres que necesitan sustituirse, el factor de expansión será 1.0 (sin expansión). Observe también que el valor MGRNO es nulo para esta fila.

```
SELECT REC2XML (1.0, 'COLATTVAL', '', DEPTNO, MGRNO, ADMRDEPT)
FROM DEPARTMENT
WHERE DEPTNO = 'D01'
```

Este ejemplo devuelve la serie VARCHAR(117) siguiente:

```

<row>
  <column name="DEPTNO">D01</column>
  <column name="MGRNO" null="true"/>
  <column name="ADMRDEPT">A00</column>
</row>

```

- Una planificación universitaria de 5 días introduce una clase llamada '&43<FIE' en una tabla llamada CL_SCHED, con un formato nuevo para la columna CLASS_CODE. Utilizando la función REC2XML, este ejemplo formatea una serie XML con estos datos de clase nuevos, exceptuando la hora final de clase.

El atributo de longitud para la llamada REC2XML (vea más abajo) con un factor de expansión de 1.0 será 128 (11 para la actividad general de '<row>' y '</row>', 21 para los nombres de columna, 75 para '<column name=', '>', '</column>' y las comillas dobles, 7 para los datos CLASS_CODE, 6 para los datos DAY y 8 para los datos STARTING). Dado que los caracteres '&' y '<' se sustituirán, no será suficiente un factor de expansión de 1.0. El atributo de longitud de la función necesitará soportar un incremento de 7 a 14 caracteres para los datos CLASS_CODE de formato nuevo.

Sin embargo, puesto que se sabe que el valor DAY no tendrá nunca más de 1 dígito de longitud, se añaden al total 5 unidades adicionales no utilizadas de longitud. Por consiguiente, la expansión sólo necesita manejar un incremento de 2. Dado que CLASS_CODE es la única columna de serie de caracteres de la lista de argumentos, éstos son los únicos datos de columna a los que se aplica el factor de expansión. Para obtener un incremento de 2 para la longitud, sería necesario un factor de expansión de 9/7 (1.2857 aproximadamente). Se utilizará un factor de expansión de 1.3.

```

SELECT REC2XML (1.3, 'COLATTVAL', 'record', CLASS_CODE, DAY, STARTING)
FROM CL_SCHED
WHERE CLASS_CODE = '&43<FIE'

```

Este ejemplo devuelve la serie VARCHAR(167) siguiente:

```

<registro>
  <column name="CLASS_CODE">&amp;43<FIE&lt;/column>
  <column name="DAY">5</column>
  <column name="STARTING">06:45:00</column>
</record>

```

- Suponga que se han añadido filas nuevas a la tabla EMP_RESUME de la base de datos de ejemplo. Las nuevas filas almacenan los resúmenes como series de XML válido. Se utiliza la *serie-formato* COLATTVAL_XML para que no se lleve a cabo la sustitución de caracteres. Ninguno de los resúmenes tiene más de 3500 caracteres de longitud. Se utiliza la consulta siguiente para seleccionar la versión XML de los resúmenes de la tabla EMP_RESUME y formatearla en un fragmento de documento XML.

REC2XML

```
SELECT REC2XML (1.0, 'COLATTVAL_XML', 'row', EMPNO, RESUME_XML)
  FROM (SELECT EMPNO, CAST(RESUME AS VARCHAR(3500)) AS RESUME_XML
        FROM EMP_RESUME
        WHERE RESUME_FORMAT = 'XML')
  AS EMP_RESUME_XML
```

Este ejemplo devuelve una fila para cada empleado que tiene un resumen en formato XML. Cada fila devuelta será una serie con el formato siguiente:

```
<row>
  <column name="EMPNO">{número de empleado}</column>
  <column name="RESUME_XML">{resumen en XML}</column>
</row>
```

Donde "{número de empleado}" es el valor EMPNO real para la columna y "{resumen en XML}" es el valor de serie de fragmento XML real que constituye el resumen.

REPEAT

►►—REPEAT—(—*expresión*—,—*expresión*—)—————►►

El esquema es SYSFUN.

Devuelve una serie de caracteres compuesta por el primer argumento repetido el número de veces especificado por el segundo argumento.

El primer argumento es una serie de caracteres o un tipo de serie binaria. Para un VARCHAR, la longitud máxima es 4.000 bytes, y para un CLOB o serie binaria, la longitud máxima es 1.048.576 bytes. El segundo argumento puede ser SMALLINT o INTEGER.

El resultado de la función es:

- VARCHAR(4000) si el primer argumento es VARCHAR (no excede de 4.000 bytes) o CHAR
- CLOB(1 M) si el primer argumento es CLOB o LONG VARCHAR
- BLOB(1 M) si el primer argumento es BLOB.

El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

Ejemplo:

- Liste la frase 'REPITA ESTO' cinco veces.
VALUES CHAR(REPEAT('REPITA ESTO', 5), 60)

Este ejemplo produce lo siguiente:

```
1
-----
REPITA ESTOREPITA ESTOREPITA ESTOREPITA ESTOREPITA ESTO
```

Tal como se ha mencionado, el resultado de la función REPEAT es VARCHAR(4000). En el ejemplo anterior, se ha utilizado la función CHAR para limitar el resultado de REPEAT a 60 bytes.

REPLACE

REPLACE

►►REPLACE(—*expresión1*—,—*expresión2*—,—*expresión3*—)◄◄

El esquema es SYSFUN.

Sustituye todas las ocurrencias de *expresión2* en la *expresión1* por la *expresión3*.

El primer argumento puede ser de cualquier tipo de serie binaria o serie de caracteres incorporado. Para un VARCHAR, la longitud máxima es 4.000 bytes y, para un CLOB o una serie binaria, la longitud máxima es 1.048.576 bytes. CHAR se convierte a VARCHAR y LONG VARCHAR se convierte a CLOB(1 M). El segundo argumento y el tercero son idénticos al primer argumento.

El resultado de la función es:

- VARCHAR(4000) si el primer, segundo y tercer argumento son VARCHAR o CHAR
- CLOB(1 M) si el primer, segundo y tercer argumento son CLOB o LONG VARCHAR
- BLOB(1 M) si el primer, segundo y tercer argumento son BLOB.

El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

Ejemplo:

- Sustituya todas las ocurrencias de la letra 'N' en la palabra 'DINING' por 'VID'.

```
VALUES CHAR (REPLACE ('DINING', 'N', 'VID'), 10)
```

Este ejemplo devuelve lo siguiente:

```
1  
-----  
DIVIDIVIDG
```

Tal como se ha mencionado, el resultado de la función REPLACE es VARCHAR(4000). En el ejemplo anterior, se ha utilizado la función CHAR para limitar el resultado de REPLACE a 10 bytes.

RIGHT

►►—RIGHT—(—*expresión1*—,—*expresión2*—)—————►►

El esquema es SYSFUN.

Devuelve una serie que consta de los *expresión2* bytes más a la derecha de *expresión1*. El valor *expresión1* se rellena con blancos por la derecha para que la subserie especificada de *expresión1* exista siempre.

El primer argumento es una serie de caracteres o un tipo de serie binaria. Para un VARCHAR, la longitud máxima es 4.000 bytes, y para un CLOB o serie binaria, la longitud máxima es 1.048.576 bytes. El segundo argumento puede ser INTEGER o SMALLINT.

El resultado de la función es:

- VARCHAR(4000) si el primer argumento es VARCHAR (no excede de 4.000 bytes) o CHAR
- CLOB(1 M) si el primer argumento es CLOB o LONG VARCHAR
- BLOB(1 M) si el primer argumento es BLOB.

El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

ROUND

►—ROUND—(—*expresión1*—,—*expresión2*—)——►

El esquema es SYSIBM.

Nota: La versión SYSFUN de la función ROUND continúa estando disponible.

La función ROUND devuelve *expresión1* redondeada a *expresión2* posiciones a la derecha de la coma decimal si *expresión2* es positiva o a la izquierda de la coma decimal si *expresión2* es cero o negativa.

Si *expresión1* es positiva, un valor de dígito de 5 o mayor es una indicación para redondear al siguiente número positivo más alto. Por ejemplo, $\text{ROUND}(3.5,0) = 4$. Si *expresión1* es negativa, un valor de dígito de 5 o mayor es una indicación para redondear al siguiente número negativo más bajo. Por ejemplo, $\text{ROUND}(-3.5,0) = -4$.

expresión1

Expresión que devuelve un valor de cualquier tipo de datos numérico incorporado.

expresión2

Expresión que devuelve un entero pequeño o grande. Cuando el valor de *expresión2* no es negativo, especifica redondear a ese número de posiciones a la derecha del separador decimal. Cuando el valor de *expresión2* es negativo, especifica redondear al valor absoluto de *expresión2* posiciones a la izquierda del separador decimal.

Si *expresión2* no es negativa, *expresión1* se redondea al valor absoluto del número de *expresión2* de posiciones a la derecha de la coma decimal. Si el valor de *expresión2* es mayor que la escala de *expresión1*, el valor no se modifica excepto si el valor de resultado tiene una precisión que es mayor por 1. Por ejemplo, $\text{ROUND}(748.58,5) = 748.58$ donde la precisión es ahora 6 y la escala sigue siendo 2.

Si *expresión2* es negativa, *expresión1* se redondea al valor absoluto de *expresión2*+1 número de posiciones a la izquierda de la coma decimal.

Si el valor absoluto de una *expresión2* negativa es mayor que el número de dígitos que hay a la izquierda de la coma decimal, el resultado es 0. Por ejemplo, $\text{ROUND}(748.58,-4) = 0$.

El tipo de datos y el atributo de longitud del resultado son iguales que el tipo de datos y el atributo de longitud del primer argumento, excepto en que la precisión se incrementa en uno si la *expresión1* es DECIMAL y la precisión es inferior a 31.

Por ejemplo, un argumento con un tipo de datos de DECIMAL(5,2) da como resultado DECIMAL(6,2). Un argumento con un tipo de datos de DECIMAL(31,2) da como resultado DECIMAL(31,2). La escala es igual que la escala del primer argumento.

Si cualquiera de los argumentos puede ser nulo o la base de datos está configurada con DFT_SQLMATHWARN establecido en YES, el resultado puede ser nulo. Si cualquiera de los argumentos es nulo, el resultado es el valor nulo.

Ejemplos:

Calcule el valor de 873.726, redondeado a 2, 1, 0, -1, -2, -3 y -4 posiciones decimales respectivamente.

```
VALUES (ROUND(873.726, 2),
        ROUND(873.726, 1),
        ROUND(873.726, 0),
        ROUND(873.726,-1),
        ROUND(873.726,-2),
        ROUND(873.726,-3),
        ROUND(873.726,-4) )
```

Este ejemplo devuelve:

1	2	3	4	5	6	7
873.730	873.700	874.000	870.000	900.000	1000.000	0.000

Realice el cálculo utilizando los números positivos y negativos.

```
VALUES (ROUND(3.5, 0),
        ROUND(3.1, 0),
        ROUNDRound(-3.1, 0),
        ROUND(-3.5,0) )
```

Este ejemplo devuelve:

1	2	3
4.0	3.0	-3.0 -4.0

RTRIM

►—RTRIM—(*—expresión-serie—*)—◄

El esquema es SYSIBM.⁴⁷

La función RTRIM elimina los blancos del final de la *expresión-serie*.

El argumento puede ser un tipo de datos CHAR, VARCHAR, GRAPHIC o VARGRAPHIC.

- Si el argumento es una serie gráfica de una base de datos DBCS o EUC, se eliminan los blancos de doble byte de cola.
- Si el argumento es una serie gráfica de una base de datos Unicode, se eliminan los blancos UCS-2 de cola.
- De lo contrario, se eliminan los blancos de un solo byte de cola.

El tipo de datos del resultado de la función es:

- VARCHAR si el tipo de datos de *expresión-serie* es VARCHAR o CHAR
- VARGRAPHIC si el tipo de datos de *expresión-serie* es VARGRAPHIC o GRAPHIC

El parámetro de longitud del tipo devuelto es el mismo que el parámetro de longitud del tipo de datos del argumento.

La longitud real del resultado para las series de caracteres es la longitud de *expresión-serie* menos el número de bytes eliminados debido a caracteres en blanco. La longitud real del resultado para series gráficas es la longitud (en número de caracteres de doble byte) de *expresión-serie* menos el número de caracteres en blanco de doble byte eliminados. Si se eliminan todos los caracteres, el resultado es una serie vacía, de longitud variable (la longitud es cero).

Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplo: Suponga que la variable del lenguaje principal HELLO está definida como CHAR(9) y tiene el valor 'Hola'.

```
VALUES RTRIM(:HELLO)
```

El resultado es 'Hola'.

47. Continúa estando disponible la versión SYSFUN de esta función con el soporte para los argumentos LONG VARCHAR y CLOB. Vea "RTRIM (esquema SYSFUN)" en la página 417 para obtener una descripción.

RTRIM (esquema SYSFUN)

►► **RTRIM** (*—expresión—*) ◀◀

El esquema es SYSFUN.

Devuelve los caracteres del argumento con los blancos de cola eliminados.

El argumento puede ser de cualquier tipo de datos de serie de caracteres incorporado. Para un VARCHAR, la longitud máxima es 4.000 bytes, y para un CLOB la longitud máxima es 1.048.576 bytes.

El resultado de la función es:

- VARCHAR(4000) si el argumento es VARCHAR (no excede de 4.000 bytes) o CHAR
- CLOB(1 M) si el argumento es CLOB o LONG VARCHAR.

El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

SECOND

SECOND

►►SECOND(—*expresión*—)◄◄

El esquema es SYSIBM.

La función SECOND devuelve la parte correspondiente a los segundos de un valor.

El argumento debe ser una hora, una indicación de fecha y hora, una duración de hora, una duración de indicación de fecha y hora o una representación válida de serie de caracteres de una hora o de una fecha y hora que no sea CLOB ni LONG VARCHAR.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Las demás reglas dependen del tipo de datos del argumento:

- Si el argumento es una hora, una indicación de fecha y hora o una representación de serie válida de una hora o de una fecha y hora:
 - El resultado es la parte correspondiente a los segundos del valor, que es un entero entre 0 y 59.
- Si el argumento es una duración de hora o una duración de indicación de fecha y hora:
 - El resultado es la parte correspondiente a los segundos del valor, que es un entero entre -99 y 99. El resultado que no es cero tiene el mismo signo que el argumento.

Ejemplos:

- Suponga que la variable del lenguaje principal TIME_DUR (decimal(6,0)) tiene el valor 153045.

```
SECOND(:TIME_DUR)
```

Devuelve el valor 45.

- Suponga que la columna RECEIVED (indicación de fecha y hora) tiene un valor interno equivalente a 1988-12-25-17.12.30.000000.

```
SECOND(RECEIVED)
```

Devuelve el valor 30.

SIGN

►►—SIGN—(*—expresión—*)——————►►

El esquema es SYSFUN.

Devuelve un indicador del signo del argumento. Si el argumento es menor que cero, devuelve -1 . Si el argumento es igual a cero, devuelve 0 . Si el argumento es mayor que cero, devuelve 1 .

El argumento puede ser de cualquier tipo de datos numérico incorporado. DECIMAL y REAL se convierten a un número de coma flotante de precisión doble para que los procese la función.

El resultado de la función es:

- SMALLINT si el argumento es SMALLINT
- INTEGER si el argumento es INTEGER
- BIGINT si el argumento es BIGINT
- de lo contrario, DOUBLE.

El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

SIN

SIN

►►SIN(*—expresión—*)◄◄

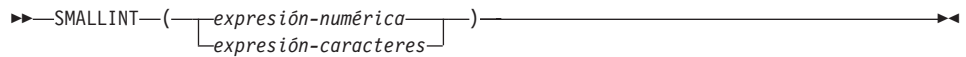
El esquema es SYSFUN.

Devuelve el seno del argumento, donde el argumento es un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo de datos numérico incorporado. Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

SMALLINT



El esquema es SYSIBM.

La función SMALLINT devuelve una representación de entero pequeño de un número o serie de caracteres en el formato de una constante de enteros pequeños.

expresión-numérica

Una expresión que devuelve un valor de cualquier tipo de datos numérico incorporado.

Si el argumento es una *expresión-numérica*, el resultado es el mismo número que sería si el argumento se asignase a una columna o variable de enteros pequeños. Si la parte correspondiente a los enteros del argumento no está dentro del rango de enteros pequeños, se produce un error. La parte correspondiente a los decimales del argumento se trunca si está presente.

expresión-caracteres

Una expresión que devuelve un valor de serie de caracteres de longitud no mayor que la longitud máxima de una constante de caracteres. Se eliminan los blancos iniciales y de cola y la serie resultante debe ajustarse a las reglas para la formación de una constante de enteros SQL (SQLSTATE 22018). Sin embargo, el valor de la constante debe estar en el rango de enteros pequeños (SQLSTATE 22003). La serie de caracteres no puede ser una serie larga.

Si el argumento es una *expresión-caracteres*, el resultado es el mismo número que sería si la constante de enteros correspondiente se asignase a una columna o variable de enteros pequeños.

El resultado de la función es un entero pequeño. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

SOUNDEX

►—SOUNDEX—(*—expresión—*)—►

El esquema es SYSFUN.

Devuelve un código de 4 caracteres que representa el sonido de las palabras del argumento. El resultado se puede utilizar para compararlo con el sonido de otras series.

El argumento puede ser una serie de caracteres de tipo CHAR o VARCHAR, cuya longitud no sea mayor que 4.000 bytes.

El resultado de la función es CHAR(4). El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

La función SOUNDEX es útil para buscar series de las que se conoce el sonido pero no su ortografía exacta. Realiza suposiciones de la manera en que el sonido de las letras y de la combinación de letras puede ayudar a buscar palabras con sonidos similares. La comparación puede realizarse directamente o pasando las series como argumentos a la función DIFFERENCE (consulte el apartado "DIFFERENCE" en la página 318).

Ejemplo:

Utilizando la tabla EMPLOYEE, busque el EMPNO y el LASTNAME del empleado cuyo apodo suena como 'Loucesy'.

```
SELECT EMPNO, LASTNAME FROM EMPLOYEE
WHERE SOUNDEX(LASTNAME) = SOUNDEX('Loucesy')
```

Este ejemplo devuelve lo siguiente:

```
EMPNO  LASTNAME
-----
000110  LUCCHESI
```

SPACE

▶▶—SPACE—(*—expresión—*)—▶▶

El esquema es SYSFUN.

Devuelve una serie de caracteres que consta de blancos con la longitud especificada por el segundo argumento.

El argumento puede ser SMALLINT o INTEGER.

El resultado de la función es VARCHAR(4000). El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

SQRT

SQRT

►►SQRT(*—expresión—*)◄◄

El esquema es SYSFUN.

Devuelve la raíz cuadrada del argumento.

El argumento puede ser de cualquier tipo de datos numérico incorporado. Se ha de convertir a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

SUBSTR

►►—SUBSTR—(—*serie*—,—*inicio*—,—*longitud*—)—►►

El esquema es SYSIBM.

La función SUBSTR devuelve una subserie de una serie.

Si la *serie* es una serie de caracteres, el resultado de la función es una serie de caracteres representada en la página de códigos del primer argumento. Si es una serie binaria, el resultado de la función es una serie binaria. Si es una serie gráfica, el resultado de la función es una serie gráfica representada en la página de códigos del primer argumento. Si cualquier argumento de la función SUBSTR puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

serie

Una expresión que especifica la serie de la que se deriva el resultado.

Si la *serie* es una serie de caracteres o una serie binaria, una subserie de *serie* es cero o más bytes contiguos de la *serie*. Si la *serie* es una serie gráfica, una subserie de *serie* es cero o más caracteres de doble byte contiguos de *serie*.

inicio

Una expresión que especifica la posición del primer byte del resultado de una serie de caracteres o de una serie binaria o la posición del primer carácter del resultado de una serie gráfica. *inicio* debe ser un entero entre 1 y la longitud o la longitud máxima de la *serie*, según si la *serie* es de longitud fija o de longitud variable (SQLSTATE 22011, si está fuera de rango). Se debe especificar como número de bytes en el contexto de la página de códigos de la base de datos, no de la página de códigos de la aplicación.

longitud

Una expresión que especifica la longitud del resultado. Si se especifica, la *longitud* debe ser un entero binario en el rango de 0 a n , donde n es igual (el atributo de longitud de la *serie*) - *inicio* + 1 (SQLSTATE 22011, si está fuera de rango).

Si *longitud* se especifica explícitamente, *serie* se rellena por la derecha con el número necesario de caracteres en blanco (de un solo byte para series de caracteres; de doble byte para series gráficas) o caracteres cero hexadecimales (para las series BLOB) para que la subserie especificada de *serie* exista siempre. El valor por omisión para la *longitud* es el número de bytes desde el byte especificado por el *inicio* hasta el último byte de la

SUBSTR

serie en el caso de la serie de caracteres o la serie binaria o el número de caracteres de doble byte del carácter especificado por el *inicio* al último carácter de la *serie* en el caso de una serie gráfica. Sin embargo, si la *serie* es una serie de longitud variable con una longitud menor que el *inicio*, el valor por omisión es cero y el resultado es la serie vacía. Se debe especificar como número de bytes en el contexto de la página de códigos de la base de datos, no de la página de códigos de la aplicación. (Por ejemplo, la columna NAME con un tipo de datos de VARCHAR(18) y un valor de 'MCKNIGHT' dará una serie vacía con SUBSTR(NAME,10).)

La Tabla 18 muestra que el tipo del resultado y la longitud de la función SUBSTR depende del tipo y los atributos de sus entradas.

Tabla 18. Tipo de datos y longitud del resultado de SUBSTR

Tipo de datos del argumento de la serie	Argumento de la longitud	Tipo de datos del resultado
CHAR(A)	constant ($l < 255$)	CHAR(l)
CHAR(A)	no especificado pero el argumento <i>inicio</i> es una constante	CHAR($A - inicio + 1$)
CHAR(A)	no es una constante	VARCHAR(A)
VARCHAR(A)	constant ($l < 255$)	CHAR(l)
VARCHAR(A)	constant ($254 < l < 32673$)	VARCHAR(l)
VARCHAR(A)	no es una constante o no se especifica	VARCHAR(A)
LONG VARCHAR	constant ($l < 255$)	CHAR(l)
LONG VARCHAR	constant ($254 < l < 4001$)	VARCHAR(l)
LONG VARCHAR	constant ($l > 4000$)	LONG VARCHAR
LONG VARCHAR	no es una constante o no se especifica	LONG VARCHAR
CLOB(A)	constant (l)	CLOB(l)
CLOB(A)	no es una constante o no se especifica	CLOB(A)
GRAPHIC(A)	constant ($l < 128$)	GRAPHIC(l)
GRAPHIC(A)	no especificado pero el argumento <i>inicio</i> es una constante	GRAPHIC($A - inicio + 1$)
GRAPHIC(A)	no es una constante	VARGRAPHIC(A)

Tabla 18. Tipo de datos y longitud del resultado de SUBSTR (continuación)

Tipo de datos del argumento de la serie	Argumento de la longitud	Tipo de datos del resultado
VARGRAPHIC(A)	constant ($l < 128$)	GRAPHIC(l)
VARGRAPHIC(A)	constant ($127 < l < 16337$)	VARGRAPHIC(l)
VARGRAPHIC(A)	no es una constante	VARGRAPHIC(A)
LONG VARGRAPHIC	constant ($l < 128$)	GRAPHIC(l)
LONG VARGRAPHIC	constant ($127 < l < 2001$)	VARGRAPHIC(l)
LONG VARGRAPHIC	constant ($l > 2000$)	LONG VARGRAPHIC
LONG VARGRAPHIC	no es una constante o no se especifica	LONG VARGRAPHIC
DBCLOB(A)	constant (l)	DBCLOB(l)
DBCLOB(A)	no es una constante o no se especifica	DBCLOB(A)
BLOB(A)	constant (l)	BLOB(l)
BLOB(A)	no es una constante o no se especifica	BLOB(A)

Si la *serie* es una serie de longitud fija, la omisión de la *longitud* equivale implícitamente a especificar `LENGTH(serie) - inicio + 1`. Si la *serie* es una serie de longitud variable, la omisión de la *longitud* equivale implícitamente a especificar 0 o `LENGTH(serie) - inicio + 1`, lo que sea mayor.

Ejemplos:

- Suponga que la variable del lenguaje principal `NAME` (`VARCHAR(50)`) tiene un valor de 'BLUE JAY' y la variable del lenguaje principal `SURNAME_POS` (`int`) tiene un valor de 6.

```
SUBSTR(:NAME, :SURNAME_POS):ehp2s
```

Devuelve el valor 'JAY'

```
SUBSTR(:NAME, :SURNAME_POS,1)
```

Devuelve el valor 'J'.

- Seleccione todas las filas de la tabla `PROJECT` para las que el nombre del proyecto (`PROJNAME`) empiece por la palabra 'OPERATION'.

SUBSTR

```
SELECT * FROM PROJECT
WHERE SUBSTR(PROJNAME,1,10) = 'OPERATION '
```

El espacio al final de la constante es necesario como preludeo de las palabras iniciales como 'OPERATIONS'.

Notas:

1. En el SQL dinámico, *serie*, *inicio* y *longitud* pueden estar representados por un marcador de parámetros (?). Si se utiliza un marcador de parámetros para la *serie*, el tipo de datos del operando será VARCHAR y el operando podrá contener nulos.
2. Aunque no se indique explícitamente en las definiciones del resultado anteriores, se deriva de esta semántica que si la *serie* es una serie de caracteres mixtos de un solo byte y de múltiples bytes, el resultado puede contener fragmentos de caracteres de múltiples bytes, según los valores de *inicio* y *longitud*. Es decir, posiblemente el resultado podría empezar en el segundo byte de un carácter de doble byte y/o finalizar en el primer byte de un carácter de doble byte. La función SUBSTR no detecta dichos fragmentos, ni proporciona ningún proceso especial si se producen.

TABLE_NAME

►►—TABLE_NAME—(—*nombreobjeto*—
 └──,—*esquemaobjeto*—┘)──►►

El esquema es SYSIBM.

La función TABLE_NAME devuelve un nombre no calificado del objeto encontrado después de que se haya resuelto cualquier cadena de seudónimos. El *nombreobjeto* especificado (y el *esquemaobjeto*) se utilizan como el punto de inicio de la resolución. Si el punto de inicio no hace referencia a un seudónimo, se devuelve el nombre no calificado del punto de inicio. El nombre resultante puede ser de una tabla, de una vista o de un objeto no definido.

nombreobjeto

Una expresión de caracteres que representa el nombre no calificado (normalmente de un seudónimo existente) que se ha de resolver. El tipo de datos de *nombreobjeto* debe ser CHAR o VARCHAR y su longitud ser mayor que 0 y menor 129 caracteres.

esquemaobjeto

Una expresión de caracteres que representa el esquema utilizado para calificar el valor del *nombreobjeto* suministrado antes de la resolución. El tipo de datos de *esquemaobjeto* debe ser CHAR o VARCHAR y su longitud ser mayor que 0 y menor 129 caracteres.

Si no se suministra el *esquemaobjeto*, se utiliza el esquema por omisión para el calificador.

El tipo de datos del resultado de la función es VARCHAR(128). Si *nombreobjeto* puede ser nulo, el resultado puede ser nulo; si *nombreobjeto* es nulo, el resultado es el valor nulo. Si *esquemaobjeto* es el valor nulo, se utiliza el nombre de esquema por omisión. El resultado es la serie de caracteres que representa un nombre no calificado. El nombre del resultado puede representar uno de los siguientes elementos:

- tabla** El valor para el *nombreobjeto* era un nombre de tabla (se devuelve el valor de entrada) o un seudónimo que se ha resuelto en la tabla cuyo nombre se devuelve.
- vista** El valor para el *nombreobjeto* era un nombre de vista (se devuelve el valor de entrada) o un seudónimo que se ha resuelto en la vista cuyo nombre se devuelve.

objeto no definido

TABLE_NAME

El valor para el *nombreobjeto* era un objeto no definido (se devuelve el valor de entrada) o un seudónimo que se ha resuelto en el objeto no definido cuyo nombre se devuelve.

Por lo tanto, si se da un valor no nulo a esta función, siempre se devuelve un valor, incluso si no existe ningún objeto con el nombre del resultado.

Ejemplos:

Vea la sección Ejemplos de “TABLE_SCHEMA” en la página 431.

TABLE_SCHEMA

```

▶▶—TABLE_SCHEMA—(—nombreobjeto—)——▶▶
                    |
                    |—esquemaobjeto—|
  
```

El esquema es SYSIBM.

La función TABLE_SCHEMA devuelve el nombre de esquema del objeto encontrado después de que se haya resuelto cualquier cadena de seudónimos. El *nombreobjeto* especificado (y el *esquemaobjeto*) se utilizan como el punto de inicio de la resolución. Si el punto de inicio no hace referencia a un seudónimo, se devuelve el nombre de esquema del punto de inicio. El nombre de esquema resultante puede ser de una tabla, de una vista o de un objeto no definido.

nombreobjeto

Una expresión de caracteres que representa el nombre no calificado (normalmente de un seudónimo existente) que se ha de resolver. El tipo de datos de *nombreobjeto* debe ser CHAR o VARCHAR y su longitud ser mayor que 0 y menor 129 caracteres.

esquemaobjeto

Una expresión de caracteres que representa el esquema utilizado para calificar el valor del *nombreobjeto* suministrado antes de la resolución. El tipo de datos de *esquemaobjeto* debe ser CHAR o VARCHAR y su longitud ser mayor que 0 y menor 129 caracteres.

Si no se suministra el *esquemaobjeto*, se utiliza el esquema por omisión para el calificador.

El tipo de datos del resultado de la función es VARCHAR(128). Si *nombreobjeto* puede ser nulo, el resultado puede ser nulo; si *nombreobjeto* es nulo, el resultado es el valor nulo. Si *esquemaobjeto* es el valor nulo, se utiliza el nombre de esquema por omisión. El resultado es la serie de caracteres que representa un nombre de esquema. El esquema del resultado puede representar el nombre de esquema para uno de los siguientes elementos:

- tabla** El valor para el *nombreobjeto* era un nombre de tabla (se devuelve la entrada o el valor por omisión de *esquemaobjeto*) o un seudónimo que se ha resuelto en una tabla para la que se devuelve el nombre de esquema.
- vista** El valor para el *nombreobjeto* era un nombre de vista (se devuelve la entrada o el valor por omisión de *esquemaobjeto*) o un seudónimo que se ha resuelto en una vista para la que se devuelve el nombre de esquema.

objeto no definido

TABLE_SCHEMA

El valor para el *nombreobjeto* era un objeto no definido (se devuelve la entrada o el valor por omisión de *esquemaobjeto*) o un seudónimo que se ha resuelto en un objeto no definido para el que se devuelve el nombre de esquema.

Por lo tanto, si se da a esta función un valor de *nombreobjeto* que no es nulo, siempre se devuelve un valor, incluso si el nombre de objeto con el nombre de esquema del resultado no existe. Por ejemplo, TABLE_SCHEMA('DEPT', 'PEOPLE') devuelve 'PEOPLE' si no se encuentra la entrada del catálogo.

Ejemplos:

- PBIRD intenta seleccionar las estadísticas para una tabla determinada de SYSCAT.TABLES utilizando un seudónimo PBIRD.A1 definido en la tabla HEDGES.T1.

```
SELECT NPAGES, CARD FROM SYSCAT.TABLES
WHERE TABNAME = TABLE_NAME ('A1')
AND TABSCHEMA = TABLE_SCHEMA ('A1')
```

Las estadísticas solicitadas para HEDGES.T1 se recuperan del catálogo.

- Seleccione las estadísticas para un objeto llamado HEDGES.X1 de SYSCAT.TABLES utilizando HEDGES.X1. Utilice TABLE_NAME y TABLE_SCHEMA ya que no se conoce si HEDGES.X1 es un seudónimo o una tabla.

```
SELECT NPAGES, CARD FROM SYSCAT.TABLES
WHERE TABNAME = TABLE_NAME ('X1','HEDGES')
AND TABSCHEMA = TABLE_SCHEMA ('X1','HEDGES')
```

Suponiendo que HEDGES.X1 sea una tabla, las estadísticas solicitadas para HEDGES.X1 se recuperan del catálogo.

- Seleccione las estadísticas para una tabla determinada de SYSCAT.TABLES utilizando un seudónimo PBIRD.A2 definido en HEDGES.T2 donde HEDGES.T2 no existe.

```
SELECT NPAGES, CARD FROM SYSCAT.TABLES
WHERE TABNAME = TABLE_NAME ('A2','PBIRD')
AND TABSCHEMA = TABLE_SCHEMA ('A2','PBIRD')
```

La sentencia devuelve 0 registros ya que no se encuentra ninguna entrada en SYSCAT.TABLES que coincida donde TABNAME = 'T2' y TABSCHEMA = 'HEDGES'.

- Seleccione el nombre calificado de cada entrada en SYSCAT.TABLES junto con el nombre de referencia final para cualquier entrada de seudónimo.

```
SELECT TABSCHEMA AS SCHEMA, TABNAME AS NAME,
TABLE_SCHEMA (BASE_TABNAME, BASE_TABSCHEMA) AS REAL_SCHEMA,
TABLE_NAME (BASE_TABNAME, BASE_TABSCHEMA) AS REAL_NAME
FROM SYSCAT.TABLES
```

La sentencia devuelve el nombre calificado para cada objeto en el catálogo y el nombre de referencia final (después de haberse resuelto el seudónimo) para cualquier entrada de seudónimo. Para todas las entradas que no son seudónimos, `BASE_TABNAME` y `BASE_TABSCHEMA` son nulos, por lo tanto las columnas `REAL_SCHEMA` y `REAL_NAME` contendrán nulos.

TAN

TAN

►►TAN(*—expresión—*)◄◄

El esquema es SYSFUN.

Devuelve la tangente del argumento, donde el argumento es un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo de datos numérico incorporado. Se ha de convertir a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

TIME

►►—TIME—(—*expresión*—)—————►►

El esquema es SYSIBM.

La función TIME devuelve una hora de un valor.

El argumento debe ser una hora, una indicación de fecha y hora o una representación de serie de caracteres válida de una hora o de una fecha y hora que no sea CLOB ni LONG VARCHAR.

El resultado de la función es una hora. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Las demás reglas dependen del tipo de datos del argumento:

- Si el argumento es una hora:
 - El resultado es dicha hora.
- Si el argumento es una indicación de fecha y hora:
 - El resultado es la parte correspondiente a la hora de la indicación de fecha y hora.
- Si el argumento es una serie de caracteres:
 - El resultado es la hora representada por la serie de caracteres.

Ejemplo:

- Seleccione todas las notas de la tabla de ejemplo IN_TRAY que se hayan recibido como mínimo una hora más tarde (de cualquier día) que la hora actual.

```
SELECT * FROM IN_TRAY
WHERE TIME(RECEIVED) >= CURRENT TIME + 1 HOUR
```

TIMESTAMP

TIMESTAMP

►—TIMESTAMP—(—*expresión*—
└, *expresión*—)——►

El esquema es SYSIBM.

La función `TIMESTAMP` devuelve una indicación de fecha y hora a partir de un valor o par de valores.

Las reglas para los argumentos dependen de si se especifica el segundo argumento.

- Si sólo se especifica un argumento:
 - Debe ser una indicación de fecha y hora, una representación válida de serie de caracteres de una fecha y hora o una serie de caracteres de longitud 14 que no sea `CLOB` ni `LONG VARCHAR`.
Una serie de caracteres de longitud 14 debe ser una serie de dígitos que represente una fecha y hora válidas con el formato *aaaaxxddhhmmss*, donde *aaaa* es el año, *xx* es el mes, *dd* es el día, *hh* es la hora, *mm* es el minuto y *ss* es los segundos.
- Si se especifican ambos argumentos:
 - El primer argumento debe ser una fecha o una representación de serie de caracteres válida de una fecha y el segundo argumento debe ser una hora o una representación de serie válida de una hora.

El resultado de la función es una indicación de fecha y hora. Si el argumento puede ser nulo, el resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

Las demás reglas dependen de si se especifica el segundo argumento:

- Si se especifican ambos argumentos:
 - El resultado es una indicación de fecha y hora, en la que el primer argumento especifica la fecha y el segundo argumento especifica la hora. La parte correspondiente a los microsegundos de la indicación de fecha y hora es cero.
- Si sólo se especifica un argumento y es una indicación de fecha y hora:
 - El resultado es esa indicación de fecha y hora.
- Si sólo se especifica un argumento y es una serie de caracteres:
 - El resultado es la indicación de fecha y hora representada por la serie de caracteres. Si el argumento es una serie de caracteres de longitud 14, la parte correspondiente a los microsegundos de la fecha y hora es cero.

Ejemplo:

- Suponga que la columna `START_DATE` (fecha) tiene un valor equivalente a 1988-12-25 y la columna `START_TIME` (hora) tiene un valor equivalente a 17.12.30.

```
TIMESTAMP(START_DATE, START_TIME)
```

Devuelve el valor '1988-12-25-17.12.30.000000'.

TIMESTAMP_ISO

TIMESTAMP_ISO

►—TIMESTAMP_ISO—(*—expresión—*)—◄

El esquema es SYSFUN.

Devuelve un valor de indicación de la hora basándose en el argumento de fecha, hora o indicación de la hora. Si el argumento es una fecha, inserta ceros para todos los elementos de hora. Si el argumento es una hora, inserta el valor de CURRENT DATE para los elementos de fecha y ceros para el elemento de fracción de hora.

El argumento debe ser una fecha, una indicación de la hora o una representación de serie de caracteres válida de una fecha o indicación de la hora que no sea CLOB ni LONG VARCHAR.

El resultado de la función es TIMESTAMP. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

TIMESTAMPDIFF

►►—TIMESTAMPDIFF—(—*expresión*—,—*expresión*—)—————►►

El esquema es SYSFUN.

Devuelve un número estimado de intervalos del tipo definido por el primer argumento, basándose en la diferencia entre dos indicaciones de la hora.

El primer argumento puede ser INTEGER o SMALLINT. Los valores válidos de intervalo (el primer argumento) son:

1	Fracciones de segundo
2	Segundos
4	Minutos
8	Horas
16	Días
32	Semanas
64	Meses
128	Trimestres
256	Años

El segundo argumento es el resultado de sustraer dos tipos de indicación de la hora y convertir el resultado a CHAR(22).

El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Se pueden utilizar las suposiciones siguientes al estimar la diferencia:

- hay 365 días en un año
- hay 30 días en un mes
- hay 24 horas en un día
- hay 60 minutos en una hora
- hay 60 segundos en un minuto

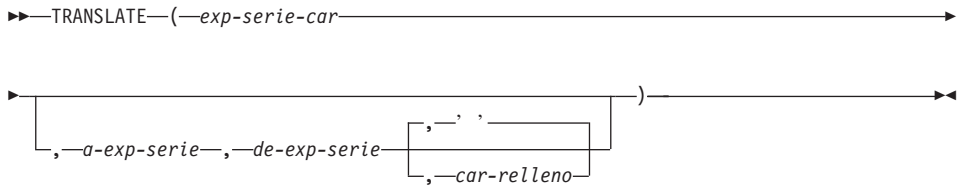
Estas suposiciones se utilizan al convertir la información del segundo argumento, que es una duración de indicación de la hora, al tipo de intervalo especificado en el primer argumento. La estimación que se devuelve puede variar en unos días. Por ejemplo, si se pide el número de días (intervalo 16) para una diferencia entre indicaciones de la hora para '1997-03-01-00.00.00' y

TIMESTAMPDIFF

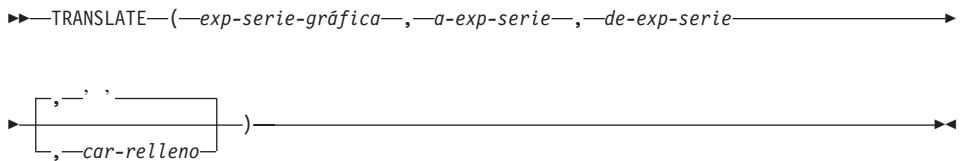
'1997-02-01-00.00.00', el resultado es 30. Esto es debido a que la diferencia entre las indicaciones de la hora es de 1 mes, por lo tanto se aplica la suposición de que hay 30 días en un mes.

TRANSLATE

expresión de serie de caracteres:



expresión de serie gráfica:



El esquema es SYSIBM.

La función TRANSLATE devuelve un valor en el que uno o varios caracteres de una expresión de serie pueden haberse convertido en otros caracteres.

El resultado de la función tiene el mismo tipo de datos y la misma página de códigos que el primer argumento. El atributo de longitud del resultado es el mismo que el del primer argumento. Si cualquier expresión especificada puede ser NULL, el resultado puede ser NULL. Si cualquier expresión especificada es NULL, el resultado será NULL.

exp-serie-car o *exp-serie-gráfica*

Una serie que se ha de convertir.

a-exp-serie

Es una serie de caracteres a la cual se convertirán algunos caracteres de *exp-serie-car*.

Si la *a-exp-serie* no está presente y el tipo de datos no es gráfico, todos los caracteres de la *exp-serie-car* estarán en mayúsculas (los caracteres de a-z se convertirán a los caracteres de A-Z y los caracteres con signos diacríticos se convertirán a sus equivalentes en mayúsculas si existen. Por ejemplo, en la página de códigos 850, é se correlaciona con É, pero ÿ no se correlaciona ya que la página de códigos 850 no incluye la Ÿ).

de-exp-serie

Es una serie de caracteres que, si se encuentra en la *exp-serie-car*, se

TRANSLATE

convertirá en el carácter correspondiente de la *a-exp-serie*. Si la *de-exp-serie* contiene caracteres duplicados, se utilizará el primero que se encuentre y los demás se pasarán por alto. Si la *a-exp-serie* es más larga que la *de-exp-serie*, se pasarán por alto los caracteres sobrantes. Si la *a-exp-serie* está presente, la *de-exp-serie* también estará presente.

exp-car-relleno

Es un solo carácter que se utilizará para rellenar la *a-exp-serie* si la *a-exp-serie* es más corta que la *de-exp-serie*. La *exp-car-relleno* debe tener un atributo de longitud de uno o se devuelve un error. Si no está presente, se tomará como un blanco de un solo byte.

Los argumentos pueden ser series de tipos de datos CHAR o VARCHAR, o series gráficas de tipo de datos GRAPHIC o VARGRAPHIC. Pueden no tener el tipo de datos LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB ni DBCLOB.

Con la *exp-serie-gráfica*, sólo el *exp-car-relleno* es opcional (si no se proporciona, se tomará el blanco de doble byte) y cada argumento, incluyendo el carácter de relleno, debe ser del tipo de datos gráfico.

El resultado es la serie que aparece después de convertir todos los caracteres de la *exp-serie-car* o la *exp-serie-gráfica* que aparece en la *de-exp-serie* al carácter correspondiente de la *a-exp-serie* o, si no existe ningún carácter correspondiente, al carácter de relleno especificado por la *exp-car-relleno*.

La página de códigos del resultado de TRANSLATE siempre es la misma página de códigos que la del primer operando, que nunca se convierte. Cada uno de los demás operandos se convierte a la página de códigos del primer operando, a menos que se defina como FOR BIT DATA (en cuyo caso no se realiza ninguna conversión).

Si los argumentos son del tipo de datos CHAR o VARCHAR, los caracteres correspondientes de la *a-exp-serie* y la *de-exp-serie* deben tener el mismo número de bytes. Por ejemplo, no es válido convertir un carácter de un solo byte a un carácter de múltiples bytes o viceversa. Se generará un error si se intenta realizarlo. La *exp-car-relleno* no debe ser el primer byte de un carácter de múltiples bytes válido o se devuelve SQLSTATE 42815. Si la *exp-car-relleno* no está presente, se tomará un blanco de un solo byte.

Si sólo se especifica *exp-serie-car*, los caracteres de un solo byte se convertirán a mayúsculas y los caracteres de múltiples bytes permanecerán sin cambios.

Ejemplos:

- Suponga que la variable del lenguaje principal SITE (VARCHAR(30)) tiene un valor de 'Hanauma Bay'.

TRANSLATE(:SITE)

Devuelve el valor 'HANAUMA BAY'.

TRANSLATE(:SITE 'j', 'B')

Devuelve el valor 'Hanauma jay'.

TRANSLATE(:SITE, 'ei', 'aa')

Devuelve el valor 'Heneume Bey'.

TRANSLATE(:SITE, 'bA', 'Bay', '%')

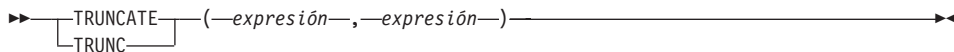
Devuelve el valor 'HAnAumA bA%'.

TRANSLATE(:SITE, 'r', 'Bu')

Devuelve el valor 'Hana ma ray'.

TRUNCATE o TRUNC

TRUNCATE o TRUNC



El esquema es SYSFUN.

Devuelve el argumento1 truncado en argumento2 posiciones a la derecha de la coma decimal. Si el argumento2 es negativo, el argumento1 se trunca al valor absoluto de argumento2 posiciones a la izquierda de la coma decimal.

El primer argumento puede ser de cualquier tipo de datos numérico incorporado. El segundo argumento tiene que ser un INTEGER o SMALLINT. DECIMAL y REAL se convierten a un número de coma flotante de precisión doble para que los procese la función.

El resultado de la función es:

- INTEGER si el primer argumento es INTEGER o SMALLINT
- BIGINT si el primer argumento es BIGINT
- DOUBLE si el primer argumento es DOUBLE, DECIMAL o DOUBLE.

El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

TYPE_ID

►►—TYPE_ID—(—*expresión*—)—————►►

El esquema es SYSIBM.

La función TYPE_ID devuelve el identificador de tipo interno del tipo de datos dinámico de la *expresión*.

El argumento debe ser un tipo estructurado definido por el usuario.⁴⁸

El tipo de datos del resultado de la función es INTEGER. Si *expresión* puede tener un valor nulo, el resultado puede ser nulo; si *expresión* tiene un valor nulo, el resultado es el valor nulo.

El valor devuelto por la función TYPE_ID no es portable a través de las bases de datos. El valor puede ser diferente aunque el esquema de tipo y el nombre de tipo del tipo de datos dinámico sean iguales. Cuando especifique el código para permitir la portabilidad, utilice las funciones TYPE_SCHEMA y TYPE_NAME para determinar el esquema de tipo y el nombre de tipo.

Ejemplos:

- Existe una jerarquía de tablas que tiene una tabla raíz EMPLOYEE de tipo EMP y una subtabla MANAGER de tipo MGR. Otra tabla ACTIVITIES incluye una columna denominada WHO_RESPONSIBLE que se define como REF(EMP) SCOPE EMPLOYEE. Para cada referencia de ACTIVITIES, visualice el identificador de tipo interno de la fila que corresponda a la referencia.

```
SELECT TASK, WHO_RESPONSIBLE->NAME,
       TYPE_ID(DEREF(WHO_RESPONSIBLE))
FROM ACTIVITIES
```

La función Deref se utiliza para devolver el objeto correspondiente a la fila.

48. Esta función no puede utilizarse como una función fuente cuando se crea una función definida por el usuario. Como acepta cualquier tipo de datos estructurado como argumento, no es necesario crear firmas adicionales para dar soporte a los diferentes tipos definidos por el usuario.

TYPE_NAME

TYPE_NAME

►—TYPE_NAME—(—*expresión*—)——►

El esquema es SYSIBM.

La función TYPE_NAME devuelve el nombre no calificado del tipo de datos dinámico de la *expresión*.

El argumento debe ser un tipo estructurado definido por el usuario.⁴⁹

El tipo de datos del resultado de la función es VARCHAR(18). Si *expresión* puede tener un valor nulo, el resultado puede ser nulo; si *expresión* tiene un valor nulo, el resultado es el valor nulo. Utilice la función TYPE_SCHEMA para determinar el nombre de esquema del nombre de tipo devuelto por TYPE_NAME.

Ejemplos:

- Existe una jerarquía de tablas que tiene una tabla raíz EMPLOYEE de tipo EMP y una subtabla MANAGER de tipo MGR. Otra tabla ACTIVITIES incluye una columna denominada WHO_RESPONSIBLE que se define como REF(EMP) SCOPE EMPLOYEE. Para cada referencia de ACTIVITIES, visualice el tipo de la fila que corresponda a la referencia.

```
SELECT TASK, WHO_RESPONSIBLE->NAME,  
       TYPE_NAME(DEREF(WHO_RESPONSIBLE)),  
       TYPE_SCHEMA(DEREF(WHO_RESPONSIBLE))  
FROM ACTIVITIES
```

La función Deref se utiliza para devolver el objeto correspondiente a la fila.

⁴⁹ Esta función no puede utilizarse como una función fuente cuando se crea una función definida por el usuario.

Como acepta cualquier tipo de datos estructurado como argumento, no es necesario crear firmas adicionales para dar soporte a los diferentes tipos definidos por el usuario.

TYPE_SCHEMA

►►—TYPE_SCHEMA—(—*expresión*—)—————►►

El esquema es SYSIBM.

La función TYPE_SCHEMA devuelve el nombre de esquema del tipo de datos dinámico de la *expresión*.

El argumento debe ser un tipo estructurado definido por el usuario.⁵⁰

El tipo de datos del resultado de la función es VARCHAR(128). Si *expresión* puede tener un valor nulo, el resultado puede ser nulo; si *expresión* tiene un valor nulo, el resultado es el valor nulo. Utilice la función TYPE_NAME para determinar el nombre de tipo asociado con el nombre de esquema devuelto por TYPE_SCHEMA.

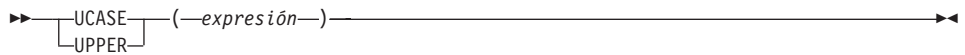
Ejemplos:

Consulte la sección Ejemplos del apartado “TYPE_NAME” en la página 446.

50. Esta función no puede utilizarse como una función fuente cuando se crea una función definida por el usuario. Como acepta cualquier tipo de datos estructurado como argumento, no es necesario crear firmas adicionales para dar soporte a los diferentes tipos definidos por el usuario.

UCASE o UPPER

UCASE o UPPER



El esquema es SYSIBM.⁵¹

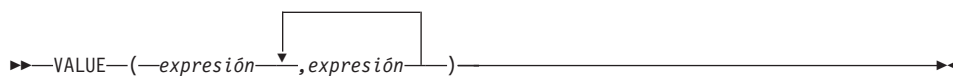
La función UCASE o UPPER es idéntica a la función TRANSLATE, excepto que sólo se especifica el primer argumento (*exp-serie-car*). Para obtener información, vea “TRANSLATE” en la página 441.

Notas:

Esta función se ha ampliado para reconocer las propiedades de minúsculas y mayúsculas de un carácter Unicode. En una base de datos Unicode, todos los caracteres Unicode se convierten correctamente a mayúsculas.

51. La versión SYSFUN de esta función sigue estando disponible para la compatibilidad con versiones superiores. Para obtener una descripción, consulte la documentación de la Versión 5.

VALUE



El esquema es SYSIBM.

La función VALUE devuelve el primer argumento que no es nulo.

VALUE es sinónimo de COALESCE. Vea “COALESCE” en la página 297 para obtener detalles.

VARCHAR

VARCHAR

De caracteres a varchar:

►► VARCHAR(*—expresión-serie-caracteres* [, *—entero*])

De fecha y hora a varchar:

►► VARCHAR(*—expresión-fechahora*)

De gráfico a varchar:

►► VARCHAR(*—expresión-serie-gráfica* [, *—entero*])

El esquema es SYSIBM.

La función VARCHAR devuelve una representación de serie de caracteres de longitud variable correspondiente a una serie de caracteres, un valor de indicación de fecha y hora o una serie gráfica (UCS-2 solamente).

El resultado de la función es una serie de longitud variable (tipo de datos VARCHAR). Si el primer argumento puede ser nulo, el resultado puede ser nulo; si el primer argumento es nulo, el resultado es el valor nulo.

De Gráfico a varchar sólo es válido para una base de datos UCS-2. En bases de datos que no sean Unicode, esto no está permitido.

De caracteres a varchar

expresión-serie-caracteres

Una expresión cuyo valor debe ser de tipo serie de caracteres, distinto de LONG VARGRAPHIC y DBCLOB, con una longitud máxima de 32.672 bytes.

entero

El atributo de longitud para la serie de caracteres de longitud variable resultante. El valor debe estar entre 0 y 32 672. Si no se especifica este argumento, la longitud del resultado es igual que la longitud del argumento.

De fecha y hora a varchar

expresión-fechahora

Una expresión cuyo valor debe ser de tipo de datos de fecha, hora o indicación de la hora.

De gráfico a varchar

expresión-serie-gráfica

Una expresión cuyo valor debe ser de tipo serie gráfica, distinto de LONG VARCHAR y DBCLOB, con una longitud máxima de 16.336 bytes.

entero

El atributo de longitud para la serie de caracteres de longitud variable resultante. El valor debe estar entre 0 y 32 672. Si no se especifica este argumento, la longitud del resultado es igual que la longitud del argumento.

Ejemplo:

- Utilizando la tabla EMPLOYEE, establezca la variable del lenguaje principal JOB_DESC (VARCHAR(8)) en el equivalente VARCHAR de la descripción del trabajo (JOB definido como CHAR(8)) para la empleada Dolores Quintana.

```
SELECT VARCHAR(JOB)
      INTO :JOB_DESC
      FROM EMPLOYEE
      WHERE LASTNAME = 'QUINTANA'
```

VARGRAPHIC

VARGRAPHIC

De carácter a vargraphic:

►► VARGRAPHIC (—*expresión-serie-caracteres*—) ►►

De gráfico a vargraphic:

►► VARGRAPHIC (—*expresión-serie-gráfica*—, —*entero*—) ►►

El esquema es SYSIBM.

La función VARGRAPHIC devuelve una representación de serie gráfica de:

- un valor de serie de caracteres, convirtiendo los caracteres de un solo byte a caracteres de doble byte
- un valor de serie gráfica, si el primer argumento es cualquier tipo de serie gráfica.

El resultado de la función es una serie gráfica de longitud variable (tipo de datos VARGRAPHIC). Si el primer argumento puede ser nulo, el resultado puede ser nulo; si el primer argumento es nulo, el resultado es el valor nulo.

De carácter a vargraphic

expresión-serie-caracteres

Una expresión cuyo valor debe ser de tipo serie de caracteres, distinto de LONG VARCHAR o CLOB, y cuya longitud máxima no debe ser mayor que 16.336 bytes.

El atributo de longitud del resultado es igual al atributo de longitud del argumento.

Suponga que S indica el valor de la *expresión-serie-caracteres*. Cada carácter de un solo byte de S se convierte en su representación de doble byte equivalente o en el carácter de sustitución de doble byte en el resultado; cada carácter de doble byte de S se correlaciona 'tal cual'. Si el primer byte de un carácter de doble byte aparece como el último byte de S, se convierte en el carácter de sustitución de doble byte. El orden secuencial de los caracteres en S se conserva.

A continuación encontrará consideraciones adicionales para la conversión.

- En una base de datos Unicode, esta función convierte la serie de caracteres de la página de códigos del operando a UCS-2. Se convierte cada carácter

del operando, incluidos los caracteres DBCS. Si se suministra el segundo argumento, especifica la longitud deseada (número de caracteres UCS-2) de la serie UCS-2 resultante.

- La conversión a elementos de código de doble byte por la función VARGRAPHIC se basa en la página de códigos del operando.
- Los caracteres de doble byte del operando no se convierten (consulte el “Apéndice O. Consideraciones acerca de EUC de japonés y de chino tradicional” en la página 1511 para ver las excepciones). El resto de caracteres se convierten a sus representaciones de doble byte correspondientes. Si no existe la representación correspondiente de doble byte, se utiliza el carácter de sustitución de doble byte para la página de códigos.
- No se genera ningún aviso ni código de error si se devuelven uno o varios caracteres de sustitución de doble byte en el resultado.

De gráfico a vargraphic

expresión-serie-gráfica

Una expresión que devuelve un valor que es una serie gráfica.

entero

El atributo de longitud para la serie gráfica de longitud variable resultante. El valor debe estar entre 0 y 16 336. Si no se especifica este argumento, la longitud del resultado es igual que la longitud del argumento.

Si la longitud de la *expresión-serie-gráfica* es mayor que el atributo de longitud del resultado, se lleva a cabo el truncamiento y se devuelve un aviso (SQLSTATE 01004) a menos que los caracteres truncados fuesen todos blancos y la *expresión-serie-gráfica* no fuese una serie larga (LONG VARGRAPHIC ni DBCLOB).

WEEK

WEEK

►►—WEEK—(*—expresión—*)——————►◄

El esquema es SYSFUN.

Devuelve la semana del año del argumento como un valor entero en el rango de 1 a 54. La semana empieza en domingo.

El argumento debe ser una fecha, una indicación de fecha y hora o una representación de serie de caracteres válida de una fecha o de una fecha y hora que no sea CLOB ni LONG VARCHAR.

El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

WEEK_ISO

►►—WEEK_ISO—(—*expresión*—)—————►►

El esquema es SYSFUN.

Devuelve la semana del año del argumento como un valor entero en el rango de 1 a 53. La semana empieza en lunes e incluye siempre 7 días. La semana 1 es la primera semana del año que contiene un jueves, lo que es equivalente a la primera semana que contiene el 4 de enero. Por consiguiente, es posible hacer que aparezca un máximo de 3 días del principio de un año en la última semana del año anterior. Y, a la inversa, pueden aparecer un máximo de 3 días del final de un año en la primera semana del año siguiente.

El argumento debe ser una fecha, una indicación de la fecha y hora o una representación de serie de caracteres válida de una fecha o de una indicación de la fecha y hora que no sea CLOB ni LONG VARCHAR.

El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplo:

La lista siguiente muestra ejemplos del resultado de WEEK_ISO y DAYOFWEEK_ISO.

DATE	WEEK_ISO	DAYOFWEEK_ISO
1997-12-28	52	7
1997-12-31	1	3
1998-01-01	1	4
1999-01-01	53	5
1999-01-04	1	1
1999-12-31	52	5
2000-01-01	52	6
2000-01-03	1	1

YEAR

YEAR

►►—YEAR—(—*expresión*—)—◄◄

El esquema es SYSIBM.

La función YEAR devuelve la parte correspondiente al año de un valor.

El argumento debe ser una fecha, una indicación de fecha y hora, una duración de fecha, una duración de indicación de fecha y hora o una representación válida de serie de caracteres de una fecha o de una fecha y hora que no sea CLOB ni LONG VARCHAR.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Las demás reglas dependen del tipo de datos del argumento especificado:

- Si el argumento es una fecha, una indicación de fecha y hora o una representación válida de serie de caracteres de una fecha o de una fecha y hora:
 - El resultado es la parte correspondiente al año del valor, que es un entero entre 1 y 9.999.
- Si el argumento es una duración de fecha o duración de indicación de fecha y hora:
 - El resultado es la parte correspondiente al año del valor, que es un entero entre -9.999 y 9.999. El resultado que no es cero tiene el mismo signo que el argumento.

Ejemplos:

- Seleccione todos los proyectos de la tabla PROJECT que se han planificado para empezar (PRSTDATE) y finalizar (PRENDATE) en el mismo año.

```
SELECT * FROM PROJECT
WHERE YEAR(PRSTDATE) = YEAR(PRENDATE)
```
- Seleccione todos los proyectos de la tabla PROJECT que se haya planificado que finalizasen en menos de un año.

```
SELECT * FROM PROJECT
WHERE YEAR(PRENDATE - PRSTDATE) < 1
```

Funciones de tabla

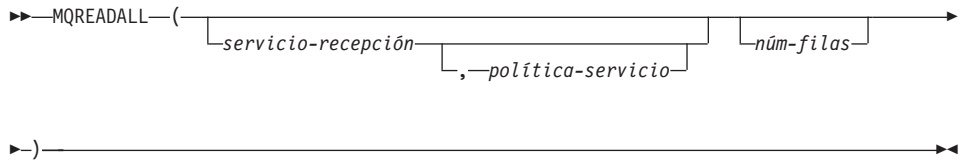
Sólo se puede utilizar una función de tabla en la cláusula FROM de una sentencia. Sin embargo, las expresiones o funciones de columnas no se pueden utilizar en una función de tabla.

Las funciones de tabla devuelven las columnas de una tabla, lo que se parece a una tabla creada por una sentencia CREATE TABLE simple.

Las funciones de tabla que siguen pueden estar calificados por el nombre de esquema.

MQREADALL

MQREADALL



El esquema es MQDB2.

La función MQREADALL devuelve una tabla que contiene los mensajes y los metadatos de mensaje de la ubicación MQSeries especificada por *servicio-recepción*, utilizando la política de calidad de servicio *política-servicio*. Al realizar esta operación no se eliminan los mensajes de la cola asociada con *servicio-recepción*.

Si se especifica *núm-filas*, se devolverá un máximo de *núm-filas* mensajes. Si no se especifica *núm-filas*, se devolverán todos los mensajes disponibles. La tabla devuelta contiene las columnas siguientes:

- MSG - columna VARCHAR(4000) que incluye el contenido del mensaje MQSeries.
- CORRELID - columna VARCHAR(24) que contiene un ID de correlación utilizado para relacionar los mensajes.
- TOPIC - columna VARCHAR(40) que contiene el tema con el que se ha publicado el mensaje, si está disponible.
- QNAME - columna VARCHAR(48) que contiene el nombre de cola donde se ha recibido el mensaje.
- MSGID - columna CHAR(24) que contiene el identificador exclusivo MQSeries asignado para este mensaje.
- MSGFORMAT - columna VARCHAR(8) que contiene el formato del mensaje, tal como lo define MQSeries. Las series típicas tienen un formato MQSTR.

servicio-recepción

Serie que contiene el destino MQSeries lógico en el cual se lee el mensaje. Si se especifica, el *servicio-recepción* debe hacer referencia a un punto de servicio definido en el archivo de depósito AMT.XML. Un punto de servicio es un punto final lógico desde el cual se envía o se recibe un mensaje. Las definiciones de punto de servicio incluyen el nombre del Gestor de colas y de la Cola de MQSeries. Consulte el manual de MQSeries Application Messaging Interface para obtener detalles

adicionales. Si no se especifica *servicio-recepción*, se utilizará DB2.DEFAULT.SERVICE. El tamaño máximo de *servicio-recepción* es de 48 bytes.

política-servicio

Serie que contiene la Política de servicio AMI de MQSeries utilizada en el manejo de este mensaje. Si se especifica, la *política-servicio* hace referencia a una Política definida en el archivo de depósito AMT.XML. Una política de servicio define un conjunto de opciones de calidad de servicio que deben aplicarse a esta operación de mensajería. Estas opciones incluyen la prioridad de mensaje y la permanencia de mensaje. Consulte el manual de MQSeries Application Messaging Interface para obtener detalles adicionales. Si no se especifica *política-servicio*, se utilizará el valor por omisión DB2.DEFAULT.POLICY. El tamaño máximo de *política-servicio* es de 48 bytes.

núm-filas

Entero positivo que contiene el número máximo de mensajes que debe devolver la función.

Ejemplos:

Ejemplo 1: En este ejemplo se reciben todos los mensajes de la cola especificada por el servicio por omisión (DB2.DEFAULT.SERVICE), utilizando la política por omisión (DB2.DEFAULT.POLICY). Los mensajes y todos los metadatos se devuelven como una tabla.

```
SELECT *
FROM table (MQREADALL()) T
```

Ejemplo 2: En este ejemplo se reciben todos los mensajes de la cabecera de la cola especificada por el servicio MYSERVICE, utilizando la política por omisión (DB2.DEFAULT.POLICY). Sólo se devuelven las columnas MSG y CORRELID.

```
SELECT T.MSG, T.CORRELID
FROM table (MQREADALL('MYSERVICE')) T
```

Ejemplo 3: En este ejemplo se lee la cabecera de la cola especificada por el servicio por omisión (DB2.DEFAULT.SERVICE), utilizando la política por omisión (DB2.DEFAULT.POLICY). Sólo se devuelven los mensajes con un CORRELID de '1234'. Se devuelven todas las columnas.

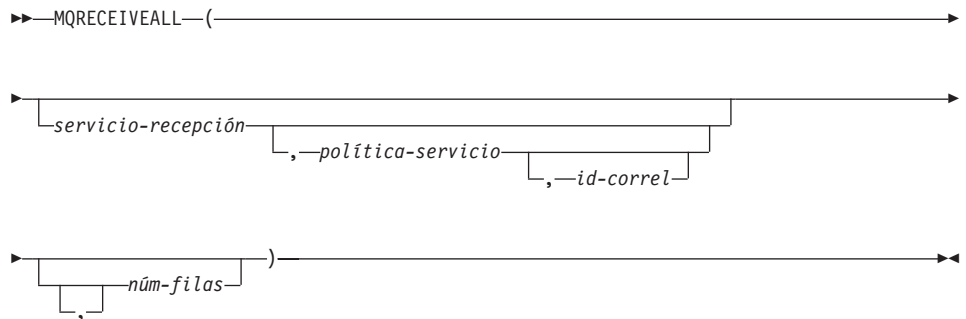
```
SELECT *
FROM table (MQREADALL()) T
WHERE T.CORRELID = '1234'
```

MQREADALL

Ejemplo 4: En este ejemplo se reciben los 10 primeros mensajes de la cabecera de la cola especificada por el servicio por omisión (DB2.DEFAULT.SERVICE), utilizando la política por omisión (DB2.DEFAULT.POLICY). Se devuelven todas las columnas.

```
SELECT *  
  FROM table (MQREADALL(10)) T
```

MQRECEIVEALL



El esquema es MQDB2.

La función MQRECEIVEALL devuelve una tabla que contiene los mensajes y los metadatos de mensaje de la ubicación MQSeries especificada por *servicio-recepción*, utilizando la política de calidad de servicio *política-servicio*. Al realizar esta operación se eliminan los mensajes de la cola asociada con *servicio-recepción*.

Si se especifica *id-correl*, sólo se devolverán los mensajes que tengan un identificador de correlación que coincida. Si no se especifica *id-correl*, se devolverá el mensaje situado en la cabecera de la cola.

Si se especifica *núm-filas*, se devolverá un máximo de *núm-filas* mensajes. Si no se especifica *núm-filas*, se devolverán todos los mensajes disponibles. La tabla devuelta contiene las columnas siguientes:

- MSG - columna VARCHAR(4000) que incluye el contenido del mensaje MQSeries.
- CORRELID - columna VARCHAR(24) que contiene un ID de correlación utilizado para relacionar los mensajes.
- TOPIC - columna VARCHAR(40) que contiene el tema con el que se ha publicado el mensaje, si está disponible.
- QNAME - columna VARCHAR(48) que contiene el nombre de cola donde se ha recibido el mensaje.
- MSGID - columna CHAR(24) que contiene el identificador exclusivo MQSeries asignado para este mensaje.
- MSGFORMAT - columna VARCHAR(8) que contiene el formato del mensaje, tal como lo define MQSeries. Las series típicas tienen un formato MQSTR.

MQRECEIVEALL

servicio-recepción

Serie que contiene el destino MQSeries lógico desde el cual se recibe el mensaje. Si se especifica, el *servicio-recepción* debe hacer referencia a un punto de servicio definido en el archivo de depósito AMT.XML. Un punto de servicio es un punto final lógico desde el cual se envía o se recibe un mensaje. Las definiciones de punto de servicio incluyen el nombre del Gestor de colas y de la Cola de MQSeries. Consulte el manual de MQSeries Application Messaging Interface para obtener detalles adicionales. Si no se especifica *servicio-recepción*, se utilizará DB2.DEFAULT.SERVICE. El tamaño máximo de *servicio-recepción* es de 48 bytes.

política-servicio

Serie que contiene la Política de servicio AMI de MQSeries utilizada en el manejo de este mensaje. Si se especifica, la *política-servicio* hace referencia a una Política definida en el archivo de depósito AMT.XML. Una política de servicio define un conjunto de opciones de calidad de servicio que deben aplicarse a esta operación de mensajería. Estas opciones incluyen la prioridad de mensaje y la permanencia de mensaje. Consulte el manual de MQSeries Application Messaging Interface para obtener detalles adicionales. Si no se especifica *política-servicio*, se utilizará el valor por omisión DB2.DEFAULT.POLICY. El tamaño máximo de *política-servicio* es de 48 bytes.

id-correl

Serie opcional que contiene un identificador de correlación asociado con este mensaje. El *id-correl* se especifica normalmente en escenarios de petición y respuesta para asociar las peticiones con las respuestas. Si no se especifica, no se especifica ningún id de correlación. El tamaño máximo de *id-correl* es de 24 bytes.

núm-filas

Entero positivo que contiene el número máximo de mensajes que debe devolver la función.

Ejemplos:

Ejemplo 1: En este ejemplo se reciben todos los mensajes de la cola especificada por el servicio por omisión (DB2.DEFAULT.SERVICE), utilizando la política por omisión (DB2.DEFAULT.POLICY). Los mensajes y todos los metadatos se devuelven como una tabla.

```
SELECT *  
FROM table (MQRECEIVEALL()) T
```

Ejemplo 2: En este ejemplo se reciben todos los mensajes de la cabecera de la cola especificada por el servicio MYSERVICE, utilizando la política por omisión (DB2.DEFAULT.POLICY). Sólo se devuelven las columnas MSG y CORRELID.

```
SELECT T.MSG, T.CORRELID  
FROM table (MQRECEIVEALL('MYSERVICE')) T
```

Ejemplo 3: En este ejemplo se reciben todos los mensajes de la cabecera de la cola especificada por el servicio "MYSERVICE", utilizando la política "MYPOLICY". Sólo se devuelven los mensajes con un CORRELID de '1234'. Sólo se devuelven las columnas MSG y CORRELID.

```
SELECT T.MSG, T.CORRELID  
FROM table (MQRECEIVEALL('MYSERVICE','MYPOLICY','1234')) T
```

Ejemplo 4: En este ejemplo se reciben los 10 primeros mensajes de la cabecera de la cola especificada por el servicio por omisión (DB2.DEFAULT.SERVICE), utilizando la política por omisión (DB2.DEFAULT.POLICY). Se devuelven todas las columnas.

```
SELECT *  
FROM table (MQRECEIVEALL(10)) T
```

SQLCACHE_SNAPSHOT

SQLCACHE_SNAPSHOT

►—SQLCACHE_SNAPSHOT—(—)——►

El esquema es SYSFUN.

SQLCACHE_SNAPSHOT devuelve los resultados de una instantánea de la antememoria de sentencias de SQL dinámicas de DB2.

La función no toma ningún argumento.

Devuelve una tabla tal como se lista más abajo. Consulte el manual *System Monitor Guide and Reference* para ver los detalles de las columnas.

Tabla 19. Nombres de columna y tipos de datos de la tabla devueltos por la función de tabla SQLCACHE_SNAPSHOT

Nombre de columna	Tipo de datos
NUM_EXECUTIONS	INTEGER
NUM_COMPILATIONS	INTEGER
PREP_TIME_WORST	INTEGER
PREP_TIME_BEST	INTEGER
INT_ROWS_DELETED	INTEGER
INT_ROWS_INSERTED	INTEGER
ROWS_READ	INTEGER
INT_ROWS_UPDATED	INTEGER
ROWS_WRITE	INTEGER
STMT_SORTS	INTEGER
TOTAL_EXEC_TIME_S	INTEGER
TOTAL_EXEC_TIME_MS	INTEGER
TOT_U_CPU_TIME_S	INTEGER
TOT_U_CPU_TIME_MS	INTEGER
TOT_S_CPU_TIME_S	INTEGER
TOT_S_CPU_TIME_MS	INTEGER
DB_NAME	VARCHAR(8)
STMT_TEXT	CLOB(64K)

Procedimientos

Un procedimiento es un programa de aplicación que se puede iniciar mediante la sentencia CALL de SQL. El procedimiento se especifica con un nombre de procedimiento, seguido de la especificación de argumentos entre paréntesis (puede que no haya argumentos).

El argumento o los argumentos de un procedimiento son valores escalares individuales, que pueden ser de tipos diferentes y pueden tener significados diferentes. Los argumentos pueden utilizarse para pasar valores en el procedimiento y recibir y/o devolver valores del procedimiento.

Los procedimientos definidos por el usuario son procedimientos que se registran en una base de datos en SYSCAT.PROCEDURES, utilizando la sentencia CREATE PROCEDURE. Con el gestor de bases de datos se proporciona un conjunto de funciones de este tipo, en un esquema llamado SYSFUN.

Los procedimientos siguientes pueden calificarse con el nombre de esquema.

Procedimientos

GET_ROUTINE_SAR

GET_ROUTINE_SAR

► GET_ROUTINE_SAR (—*sarblob*—, —*tipo*—, —*serie_nombre_rutina*—) ◄

El esquema es SYSFUN.

El procedimiento GET_ROUTINE_SAR recupera la información necesaria para instalar la misma rutina en otro servidor de bases de datos que se ejecute en el mismo nivel en el mismo sistema operativo. La información se recupera en una serie BLOB individual que representa un archivo archivador SQL. La persona que invoca el procedimiento GET_ROUTINE_SAR debe tener autorización DBADM.

sarblob

Argumento de salida de tipo BLOB(3M) que incluye el contenido del archivo SAR de la rutina.

tipo

Argumento de entrada de tipo CHAR(2) que especifica el tipo de rutina, utilizando uno de los valores siguientes:

- **P** para un procedimiento.
- **SP** para el nombre específico de un procedimiento.

serie_nombre_rutina

Argumento de entrada de tipo VARCHAR(257) que especifica un nombre calificado de la rutina. Si no se especifica ningún nombre de esquema, el valor por omisión es CURRENT SCHEMA cuando se procesa la rutina.

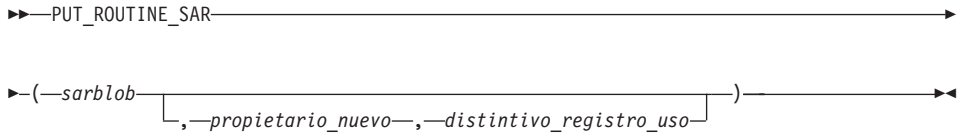
Nota: La *serie_nombre_rutina* no puede incluir el carácter de comillas (").

El nombre calificado de la rutina se utiliza para determinar qué rutina se debe recuperar. La rutina que se encuentre debe ser una rutina SQL o se producirá un error (SQLSTATE 428F7). Si no se utiliza un nombre específico, puede producirse más de una rutina y se producirá un error (SQLSTATE 42725). Si ocurre esto, para obtener la rutina se deberá utilizar el nombre específico de la misma.

El archivo SAR debe incluir un archivo de enlace que puede no estar disponible en el servidor. Si el archivo de enlace no se puede encontrar y almacenar en el archivo SAR, se producirá un error (SQLSTATE 55045).

PUT_ROUTINE_SAR

PUT_ROUTINE_SAR



El esquema es SYSFUN.

El procedimiento PUT_ROUTINE_SAR pasa el archivo necesario para crear una rutina SQL en el servidor y, a continuación, define la rutina. La persona que invoca el procedimiento PUT_ROUTINE_SAR debe tener autorización DBADM.

sarblob

Argumento de entrada de tipo BLOB(3M) que incluye el contenido del archivo SAR de la rutina.

propietario_nuevo

Argumento de entrada de tipo VARCHAR(128) que contiene un nombre-autorización utilizado para la comprobación de autorización de la rutina. El *propietario-nuevo* debe tener los privilegios necesarios para que se defina la rutina. Si no se especifica *propietario-nuevo*, se utilizará el nombre-autorización del definidor de rutina original.

distintivo_registro_uso

Argumento de entrada de tipo INTEGER que indica si los registros especiales CURRENT SCHEMA y CURRENT PATH se utilizan para definir la rutina. Si no se utilizan los registros especiales, los valores para el esquema por omisión y la vía de acceso de SQL son los valores utilizados cuando se ha definido originalmente la rutina. Valores posibles para *distintivo-registro-uso*:

- 0 No utilizar los registros especiales del entorno actual
- 1 Utilizar los registros especiales CURRENT SCHEMA y CURRENT PATH.

Si el valor es 1, se utiliza CURRENT SCHEMA para los nombres de objeto no calificados en la definición de rutina (incluido el nombre de la rutina) y CURRENT PATH se utiliza para resolver rutinas no calificadas y tipos de datos en la definición de rutina. Si no se especifica *distintivo-registro-uso*, el comportamiento es el mismo que si se hubiera especificado un valor de 0.

Procedimientos

La información de identificación contenida en *sarblob* se comprueba para confirmar que las entradas son apropiadas para el entorno porque, de lo contrario, se produce un error (SQLSTATE 55046). Entonces el procedimiento `PUT_ROUTINE_SAR` utiliza el contenido de *sarblob* para definir la rutina en el servidor.

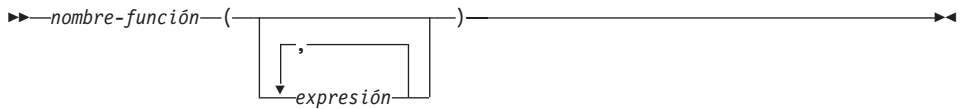
El contenido del argumento *sarblob* se extrae en los archivos independientes que forman el archivo archivador SQL. La biblioteca compartida y los archivos de enlace se graban en archivos de un directorio temporal. El entorno se establece de forma tal que el proceso de sentencia de definición de rutina esté informado de que la compilación y el enlace no son necesarios y que la ubicación de la biblioteca compartida y los archivos de enlace está disponible. Entonces se utiliza el contenido del archivo DDL para ejecutar dinámicamente la sentencia de definición de rutina.

Nota: No se puede instalar concurrentemente más de un procedimiento bajo un esquema determinado.

El proceso de esta sentencia puede producir los mismos errores que la ejecución de la sentencia de definición de rutina utilizando otras interfaces. Durante el proceso de definición de rutina, se detecta la presencia de la biblioteca compartida y de los archivos de enlace y se saltan los pasos de precompilación, compilación y enlace. El archivo de enlace se utiliza durante el proceso de enlace y el contenido de ambos archivos se copia en el directorio usual para una rutina SQL.

Nota: Si no se puede ejecutar satisfactoriamente una operación `GET ROUTINE` o `PUT ROUTINE` (o su procedimiento correspondiente), se devolverá siempre un error (SQLSTATE 38000), junto con texto de diagnóstico que proporciona información acerca de la causa de la anomalía. Por ejemplo, si el nombre de procedimiento proporcionado a `GET ROUTINE` no identifica un procedimiento SQL, se devolverá el texto de diagnóstico "100, 02000", donde "100" y "02000" son `SQLCODE` y `SQLSTATE`, respectivamente, que identifican la causa del problema. En este ejemplo, `SQLCODE` y `SQLSTATE` indican que la fila especificada para el nombre de procedimiento proporcionado no se ha encontrado en las tablas del catálogo.

Funciones definidas por el usuario



Las funciones definidas por el usuario son extensiones o adiciones a las funciones incorporadas existentes del lenguaje SQL. Una función definida por el usuario puede ser una función escalar, que devuelve un solo valor cada vez que se invoca, una función de columna, a la que se pasa un conjunto de valores similares y devuelve un solo valor para el conjunto, una función de fila, que devuelve una fila, o una función de tabla, que devuelve una tabla. Observe que una UDF puede ser una función de columna sólo cuando deriva de una función de columna existente.

Una función escalar o función de columna definida por el usuario que esté registrada en la base de datos puede referenciarse en el mismo contexto donde pueda aparecer una función incorporada cualquiera.

Una función de tabla definida por el usuario registrada en la base de datos sólo puede referenciarse en la cláusula FROM de una sentencia SELECT, tal como se describe en “cláusula-from” en la página 478.

Una función de fila definida por el usuario sólo puede referenciarse implícitamente cuando está registrada como función de transformación para un tipo definido por el usuario.

Se hace referencia a una función definida por el usuario mediante un nombre de función calificado o no calificado, seguido de paréntesis que encierran los argumentos de la función (si los hay).

Los argumentos de la función deben corresponderse en número y posición con los parámetros especificados en la función definida por el usuario tal como se ha registrado con la base de datos. Además, los argumentos deben ser de tipos de datos promocionables a los tipos de datos de los parámetros definidos correspondientes. (consulte el apartado “CREATE FUNCTION” en la página 695).

El resultado de la función es el especificado en la cláusula RETURNS que se especificó al registrar la función definida por el usuario. La cláusula RETURNS determina si una función es una función de tabla o no.

Si estaba especificada la cláusula RETURNS NULL ON NULL INPUT (o ésta tomó el valor por omisión) cuando se registró la función, entonces si cualquier

Funciones definidas por el usuario

argumento es nulo, el resultado será nulo. En las funciones de tabla, esto se interpreta como una tabla de retorno sin filas (una tabla vacía).

Existe una colección de funciones definidas por el usuario proporcionadas en el esquema SYSFUN (consulte la Tabla 15 en la página 236).

Ejemplos:

- Suponga que una función escalar llamada ADDRESS se ha escrito para extraer la dirección personal de un resumen en formato de script. La función ADDRESS espera un argumento CLOB y devuelve VARCHAR(4000). El siguiente ejemplo ilustra la invocación de la función ADDRESS.

```
SELECT EMPNO, ADDRESS(RESUME) FROM EMP_RESUME  
WHERE RESUME_FORMAT = 'SCRIPT'
```

- Suponga una tabla T2 con una columna A numérica y la función ADDRESS descrita en el ejemplo anterior. El ejemplo siguiente ilustra un intento de invocar la función ADDRESS con un argumento incorrecto.

```
SELECT ADDRESS(A) FROM T2
```

Se genera un error (SQLSTATE 42884) ya que no hay ninguna función con un nombre que coincida y con un parámetro promocionable del argumento.

- Suponga que se ha escrito una función de tabla WHO para que devuelva información acerca de las sesiones de la máquina servidora que estaba activa en el momento de ejecutar la sentencia. El ejemplo siguiente ilustra la invocación de WHO en una cláusula FROM (palabra clave TABLE con variable de correlación obligatoria).

```
SELECT ID, START_DATE, ORIG_MACHINE  
FROM TABLE( WHO() ) AS QQ  
WHERE START_DATE LIKE 'MAY%'
```

Los nombres de columna de la tabla WHO() se definen en la sentencia CREATE FUNCTION.

Capítulo 5. Consultas

Una *consulta* especifica una tabla resultante.

Una consulta es un componente de algunas sentencias de SQL. Las tres formas de una consulta son:

- subselección
- selección completa
- sentencia-select.

Hay otra sentencia de SQL que puede utilizarse para recuperar como mucho una sola fila que se describe en el apartado “SELECT INTO” en la página 1145.

Autorización

El ID de autorización de la sentencia debe tener como mínimo uno de los siguientes privilegios o autorizaciones para cada tabla o vista a las que la consulta haga referencia:

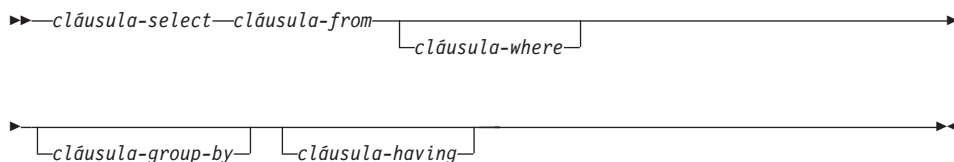
- Autorización SYSADM o DBADM
- Privilegio CONTROL
- Privilegio SELECT.

No se comprueban los privilegios de grupo en las consultas contenidas en sentencias de SQL estáticas.

En los apodos referenciados en una consulta, no se tienen en cuenta los privilegios de la base de datos federada. Se aplican los requisitos de autorización de la fuente de datos para la tabla o vista a la que hace referencia el apodo cuando se procesa la consulta. El ID de autorización de la sentencia puede estar correlacionado con un ID de autorización remoto diferente.

subselección

subselección



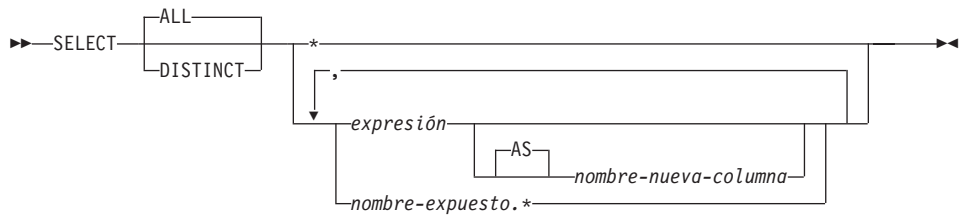
La *subselección* es un componente de la selección completa.

Una subselección especifica una tabla resultante que deriva de las tablas, vistas o apodos identificados en la cláusula FROM. La derivación puede describirse como una secuencia de operaciones en las que el resultado de cada operación es la entrada de la siguiente. (Es la única manera de describir la subselección. El método utilizado para realizar la derivación puede ser bastante diferente de esta descripción.)

Las cláusulas de la subselección se procesan en el orden siguiente:

1. Cláusula FROM
2. Cláusula WHERE
3. Cláusula GROUP BY
4. Cláusula HAVING
5. Cláusula SELECT.

cláusula-select



La cláusula SELECT especifica las columnas de la tabla resultante final. Los valores de columna los genera la aplicación de la *lista de selección* a R. La lista de selección son los nombres o expresiones especificados en la cláusula SELECT y R es el resultado de la operación anterior de la subselección. Por ejemplo, si las únicas cláusulas especificadas con SELECT, FROM y WHERE, R es el resultado de la cláusula WHERE.

ALL

Retiene todas las filas de la tabla resultante final y no elimina los duplicados redundantes. Éste es el valor por omisión.

DISTINCT

Elimina todas las filas excepto una de los juegos de filas duplicadas de la tabla resultante final. Si se utiliza DISTINCT, ninguna columna de tipo serie de la tabla resultante puede tener una longitud máxima mayor que 255 bytes, y ninguna columna puede ser un tipo LONG VARCHAR, LONG VARGRAPHIC, DATALINK, LOB, un tipo diferenciado de cualquiera de estos tipos ni un tipo estructurado. DISTINCT puede utilizarse más de una vez en una subselección. Esto incluye SELECT DISTINCT, la utilización de DISTINCT en una función de columna de la lista de selección o la cláusula HAVING y las subconsultas de la subselección.

Dos filas sólo son duplicadas una de la otra si cada valor de la primera es igual al valor correspondiente de la segunda. Para la determinación de duplicados, dos valores nulos se consideran iguales.

Notación de lista de selección:

- * Representa una lista de nombres que identifican las columnas de la tabla R. El primer nombre de la lista identifica la primera columna de R, el segundo nombre identifica la segunda columna de R y así sucesivamente.

La lista de nombres se establece cuando se enlaza el programa que contiene la cláusula SELECT. Por lo tanto, * (el asterisco) no identifica ninguna columna que se haya añadido a la tabla después de que se haya enlazado la sentencia que contiene la referencia a la tabla.

cláusula-select

expresión

Especifica los valores de una columna del resultado. Puede ser cualquier expresión del tipo descrito en el Capítulo 3, pero las expresiones utilizadas normalmente incluyen nombres de columna. Cada nombre de columna utilizado en la lista de selección debe identificar sin ambigüedades una columna R.

nuevo-nombre-columna o **AS** *nuevo-nombre-columna*

Nombra o cambia el nombre de la columna del resultado. El nombre no debe estar calificado y no tiene que ser exclusivo. El uso subsiguiente del nombre-columna está limitado en lo siguiente:

- Un nuevo-nombre-columna especificado en la cláusula AS se puede utilizar en la cláusula-order-by, siempre que sea exclusivo.
- Un nuevo-nombre-columna especificado en la cláusula AS de la lista de selección no se puede utilizar en ninguna otra cláusula de la subselección (cláusula-where, cláusula-group-by o cláusula-having).
- Un nuevo-nombre-columna especificado en la cláusula AS no se puede utilizar en la cláusula-update.
- Un nuevo-nombre-columna especificado en la cláusula AS se conoce fuera de la selección completa de las expresiones de tabla anidadas, las expresiones de tablas comunes y CREATE VIEW.

*nombre.**

Representa la lista de nombres que identifican las columnas de la tabla resultante identificada por *nombre-expuesto*. El *nombre-expuesto* puede ser un nombre de tabla, un nombre de vista, un apodo o un nombre de correlación, y debe designar una tabla, una vista o un apodo especificado en la cláusula FROM. El primer nombre de la lista identifica la primera columna de la tabla, vista o apodo, el segundo nombre de la lista identifica la segunda columna de la tabla, vista o apodo, y así sucesivamente.

La lista de nombres se establece cuando se enlaza la sentencia que contiene la cláusula SELECT. Por lo tanto, * no identifica ninguna columna que se haya añadido a la tabla después de enlazar la sentencia.

El número de columnas del resultado de SELECT es igual que el número de expresiones de la forma operativa de la lista de selección (es decir, la lista establecida cuando se ha preparado la sentencia) y no puede exceder de 500.

Límites en las columnas de serie

Para ver las limitaciones en la lista de selección, consulte el apartado “Restricciones aplicables a la utilización de series de caracteres de longitud variable” en la página 87.

Aplicación de la lista de selección

Algunos resultados de aplicar la lista de selección en R dependen de si se utiliza GROUP BY o HAVING o no. Los resultados se describen en dos listas separadas:

Si se utiliza GROUP BY o HAVING:

- Una expresión X (no una función de columna) utilizada en la lista de selección debe contener una cláusula GROUP BY con:
 - una *expresión-agrupación* en la cual cada nombre-columna identifique sin ambigüedades una columna de R (consulte el apartado “Cláusula group-by” en la página 487), o bien,
 - cada columna de R a la que X haga referencia como una *expresión-agrupación* separada.
- La lista de selección se aplica a cada grupo de R y el resultado contiene tantas filas como grupos haya en R. Cuando la lista de selección se aplica a un grupo de R, este grupo es el origen de los argumentos de las funciones de columna de la lista de selección.

Si no se utilizan ni GROUP BY ni HAVING:

- La lista de selección no debe incluir ninguna función de columna o cada *nombre-columna* de la lista de selección debe estar especificado en una función de columna o debe ser una referencia de columna correlacionada.
- Si la selección no incluye funciones de columna, la lista de selección se aplica a cada fila de R y el resultado contiene tantas filas como el número de filas en R.
- Si la lista de selección es una lista de funciones de columna, R es la fuente de los argumentos de las funciones y el resultado de aplicar la lista de selección es una fila.

En cualquier caso la columna n del resultado contiene los valores especificados por la aplicación de la expresión n en la forma operativa de la lista de selección.

Atributos nulos de las columnas del resultado: Las columnas del resultado no permiten valores nulos si se derivan de:

- Una columna que no permite valores nulos
- Una constante
- La función COUNT o COUNT_BIG
- Una variable del lenguaje principal que no tiene una variable indicadora
- Una función o expresión escalar que no incluye un operando que permita nulos.

Las columnas del resultado permiten valores nulos si se derivan de:

cláusula-select

- Cualquier función de columna excepto COUNT o COUNT_BIG
- Una columna que permite valores nulos
- Una función o expresión escalar que incluye un operando que permite nulos
- Una función NULLIF con argumentos que contienen valores iguales.
- Una variable del lenguaje principal que contiene una variable indicadora.
- Un resultado de una operación de conjuntos si como mínimo uno de los elementos correspondientes de la lista de selección admite nulos.
- Una expresión aritmética o columna de vista que se deriva de una expresión aritmética y la base de datos está configurada con DFT_SQLMATHWARN establecido en "yes"
- Una operación de desreferencia.

Nombres de las columnas del resultado:

- Si se especifica la cláusula AS, el nombre de la columna del resultado es el nombre especificado en esta cláusula.
- Si no se especifica la cláusula AS y la columna del resultado se deriva de una columna, el nombre de columna del resultado es el nombre no calificado de dicha columna.
- Si no se especifica la cláusula AS y la columna del resultado se deriva mediante una operación de desreferencia, el nombre de columna del resultado es el nombre no calificado de la columna de destino de la operación de desreferencia.
- Todos los demás nombres de columna del resultado carecen de nombre.⁵²

Tipos de datos de las columnas del resultado: Cada columna del resultado de SELECT adquiere un tipo de datos de la expresión de la que se deriva.

Cuando la expresión es ...	El tipo de datos de la columna del resultado es ...
el nombre de cualquier columna numérica	el mismo que el tipo de datos de la columna, con la misma precisión y escala que para las columnas DECIMAL.
una constante de enteros	INTEGER
una constante decimal	DECIMAL, con la precisión y escala de la constante.
una constante de coma flotante	DOUBLE

52. El sistema asigna números temporales (como series de caracteres) a estas columnas.

Quando la expresión es ...	El tipo de datos de la columna del resultado es ...
el nombre de cualquier variable numérica	el mismo que el tipo de datos de la variable, con la misma precisión y escala que para las variables DECIMAL.
una expresión	Para ver una descripción de los atributos del tipo de datos, vea "Expresiones" en la página 176.
cualquier función	(consulte el Capítulo 4 para determinar el tipo de datos del resultado.)
una constante hexadecimal que representa n bytes	VARCHAR(n). La página de códigos es la página de códigos de la base de datos.
el nombre de cualquier columna de serie	el mismo que el tipo de datos de la columna, con el mismo atributo de longitud.
el nombre de cualquier variable de serie.	el mismo que el tipo de datos de la variable, con el mismo atributo de longitud. Si el tipo de datos de la variable no es idéntico a un tipo de datos SQL (por ejemplo, una serie terminada en NUL en C), la columna del resultado es una serie de longitud variable.
una constante de serie de caracteres de longitud n	VARCHAR(n)
una constante de serie gráfica de longitud n	VARGRAPHIC(n)
el nombre de una columna de indicación de fecha y hora	el mismo tipo de datos de la columna.
el nombre de una columna de tipo definido por el usuario	el mismo que el tipo de datos de la columna
el nombre de una columna de tipo de referencia	el mismo que el tipo de datos de la columna

cláusula-from

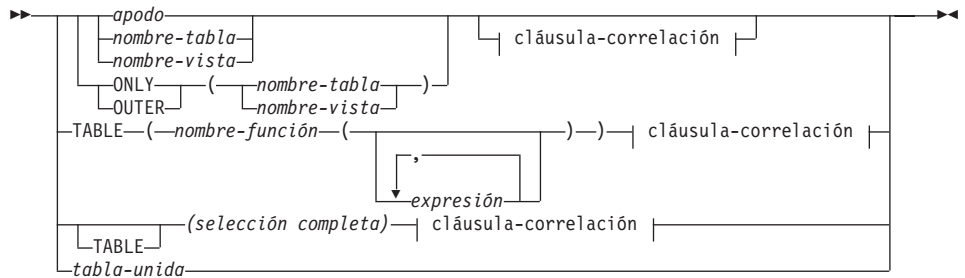
cláusula-from



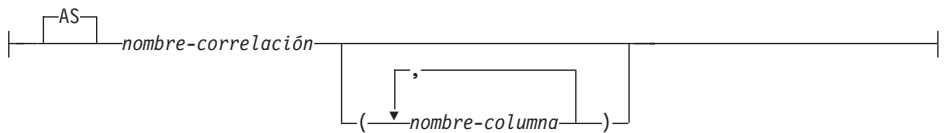
La cláusula FROM especifica una tabla resultante intermedia.

Si se especifica una referencia-tabla, la tabla resultante intermedia es simplemente el resultado de dicha referencia-tabla. Si se especifica más de una referencia-tabla, la tabla resultante intermedia consta de todas las combinaciones posibles de las filas de las referencias-tabla especificadas (el producto cartesiano). Cada fila del resultado es una fila de la primera referencia-tabla concatenada con una fila de la segunda referencia-tabla concatenada a su vez con una fila de la tercera, etcétera. El número de filas del resultado es el producto del número de filas de todas las referencias-tabla individuales. Para obtener una descripción de *referencia-tabla*, consulte el apartado “referencia-tabla” en la página 479.

referencia-tabla



cláusula-correlación:



Cada *nombre-tabla*, *nombre-vista* o *apodo* especificado como referencia-tabla debe identificar una tabla, vista o apodo existente del servidor de aplicaciones o el *nombre-tabla* de una expresión de tabla común (vea “expresión-común-tabla” en la página 519) definida antes de la selección completa que contiene la referencia-tabla. Si *nombre-tabla* hace referencia a una tabla con tipo, el nombre indica UNION ALL de la tabla con todas sus subtablas y solamente con las columnas de *nombre-tabla*. De manera similar, si *nombre-vista* hace referencia a una vista con tipo, el nombre indica UNION ALL de la vista con todas sus subvistas y solamente con las columnas de *nombre-vista*.

El uso de `ONLY(nombre-tabla)` u `ONLY(nombre-vista)` significa que no se incluyen las filas de las subtablas o subvistas correspondientes. Si el *nombre-tabla* utilizado con `ONLY` no tiene subtablas, `ONLY(nombre-tabla)` equivale a especificar *nombre-tabla*. Si el *nombre-vista* utilizado con `ONLY` no tiene subvistas, `ONLY(nombre-vista)` equivale a especificar *nombre-vista*.

El uso de `OUTER(nombre-tabla)` u `OUTER(nombre-vista)` representa una tabla virtual. Si el *nombre-tabla* o el *nombre-vista* que se utilice con `OUTER` no tiene subtablas o subvistas, especificar `OUTER` equivale a no especificar `OUTER`. `OUTER(nombre-tabla)` se deriva de *nombre-tabla* de la manera siguiente:

- En las columnas, se incluyen las columnas de *nombre-tabla* seguidas de las columnas adicionales que ha introducido cada una de sus subtablas (si existen). Las columnas adicionales se añaden a la derecha atravesando la

referencia-tabla

jerarquía de las subtablas por orden de importancia. Se atraviesan las subtablas que tienen un padre común en el orden de creación de sus tipos.

- En las filas, se incluyen todas las filas de *nombre-tabla* y todas las filas de sus subtablas. Se devuelven valores nulos para las columnas que no están en la subtabla para la fila.

Los puntos anteriores también se aplican a OUTER(*nombre-vista*) si se sustituye *nombre-tabla* por *nombre-vista* y subtabla por subvista.

El uso de ONLY o OUTER requiere el privilegio SELECT en cada subtabla de *nombre-tabla* o subvista de *nombre-vista*.

Cada *nombre-función*, junto con los tipos de sus argumentos, especificado como una referencia a tabla, debe resolverse en una función de tabla existente en el servidor de aplicaciones.

Una selección completa entre paréntesis seguida de un nombre de correlación se llama una *expresión de tabla anidada*.

Una *tabla-unida* especifica un conjunto resultante intermedio que es el resultado de una o varias operaciones de unión. Para obtener información, vea “tabla-unida” en la página 483.

Los nombres expuestos de todas las referencias a tabla deben ser exclusivos. Un nombre expuesto es:

- Un *nombre-correlación*,
- Un *nombre-tabla* que no va seguido de un *nombre-correlación*,
- Un *nombre-vista* que no va seguido de un *nombre-correlación*,
- Un *apodo* que no va seguido de un *nombre-correlación*,
- Un *nombre-seudónimo* que no va seguido de un *nombre-correlación*.

Cada *nombre-correlación* se define como un designador de la referencia inmediatamente precedente de *nombre-tabla*, *nombre-vista*, *apodo*, *nombre-función* o expresión de tabla anidada. Una referencia calificada a una columna para una tabla, vista, función de tabla o expresión de tabla anidada debe utilizar el nombre expuesto. Si se especifica dos veces el mismo nombre de tabla, vista o apodo, como mínimo una especificación debe ir seguida de un *nombre-correlación*. El *nombre-correlación* se utiliza para calificar las referencias a las columnas de la tabla, vista o apodo. Cuando se especifica un *nombre-correlación*, también se pueden especificar *nombres-columna* para asignar nombres a las columnas de la referencia a *nombre-tabla*, *nombre-vista*, *apodo*, *nombre-función* o expresión de tabla anidada. Para obtener más información, vea “Nombres de correlación” en la página 144.

En general, las funciones de tabla y las expresiones de tabla anidadas pueden especificarse en cualquier cláusula-from. Las columnas de las funciones de tabla y las expresiones de tabla anidadas pueden hacerse referencia en la lista de selección y en el resto de la subselección utilizando el nombre de correlación que debe especificarse. El ámbito de este nombre de correlación es el mismo que los nombres de correlación para otros nombres de tabla, vista o apodo de la cláusula FROM. Una expresión de tabla anidada puede utilizarse:

- en el lugar de una vista para evitar la creación de la vista (cuando no es necesario el uso general de la vista)
- cuando la tabla resultante deseada se basa en variables del lenguaje principal.

Referencias a funciones de tabla

En general, se puede hacer referencia a una función de tabla junto a los valores de sus argumentos en la cláusula FROM de una sentencia SELECT exactamente de la misma manera que una tabla o una vista. Sin embargo, se aplican algunas consideraciones especiales.

- Nombres de columna de función de tabla

A menos que se proporcionen nombres de columna alternativos a continuación del *nombre-correlación*, los nombres de columna de la función de tabla son los especificados en la cláusula RETURNS de la sentencia CREATE FUNCTION. Es análogo a los nombres de las columnas de una tabla, que se definen en la sentencia CREATE TABLE. Vea “CREATE FUNCTION (Tabla externa)” en la página 723 o “CREATE FUNCTION (SQL, escalar, de tabla o de fila)” en la página 761 para obtener detalles sobre la creación de una función de tabla.

- Resolución de una función de tabla

Los argumentos especificados en una referencia de función de tabla, junto con el nombre de la función, se utilizan por un algoritmo llamado *resolución de función* para determinar la función exacta que se utiliza. Esta operación no es diferente de lo que ocurre con las demás funciones (por ejemplo, en las funciones escalares), utilizadas en una sentencia. La resolución de función se explica el apartado “Resolución de funciones” en la página 162.

- Argumentos de función de tabla

Como en los argumentos de funciones escalares, los argumentos de función de tabla pueden ser en general cualquier expresión SQL válida. Por lo tanto, los ejemplos siguientes son de sintaxis válida:

```
Ejemplo 1: SELECT c1
            FROM TABLE( tf1('Zachary') ) AS z
            WHERE c2 = 'FLORIDA';
```

```
Ejemplo 2: SELECT c1
            FROM TABLE( tf2 (:hostvar1, CURRENT DATE) ) AS z;
```

```
Ejemplo 3: SELECT c1
```

referencia-tabla

```
FROM t
WHERE c2 IN
      (SELECT c3 FROM
       TABLE( tf5(t.c4) ) AS z   -- referencia correlacionada
      )                          -- a cláusula FROM anterior
```

Referencias correlacionadas en referencias-tabla

Las referencias correlacionadas se pueden utilizar en expresiones de tabla anidadas o como argumentos para funciones de tabla. La regla básica que se aplica para ambos casos es que la referencia correlacionada debe ser de una *referencia-tabla* de un nivel superior en la jerarquía de subconsultas. Esta jerarquía incluye las referencias-tabla que ya se han resuelto en el proceso de izquierda a derecha de la cláusula FROM. En las expresiones de tabla anidadas, la palabra clave TABLE debe aparecer antes de la selección completa. Por lo tanto, los ejemplos siguientes son de sintaxis válida:

Ejemplo 1: **SELECT** t.c1, z.c5
FROM t, **TABLE**(tf3(t.c2)) **AS** z -- t precede a tf3 en FROM
WHERE t.c3 = z.c4; -- por tanto t.c2 es conocido

Ejemplo 2: **SELECT** t.c1, z.c5
FROM t, **TABLE**(tf4(2 * t.c2)) **AS** z -- t precede a tf3 en FROM
WHERE t.c3 = z.c4; -- por tanto t.c2 es conocido

Ejemplo 3: **SELECT** d.deptno, d.deptname,
empinfo.avgsal, empinfo.empcount
FROM department d,
TABLE (**SELECT** **AVG**(e.salary) **AS** avgsal,
COUNT(*) **AS** empcount
FROM employee e -- department precede y
WHERE e.workdept=d.deptno -- se especifica TABLE
) AS empinfo; -- por tanto d.deptno es
 -- conocido

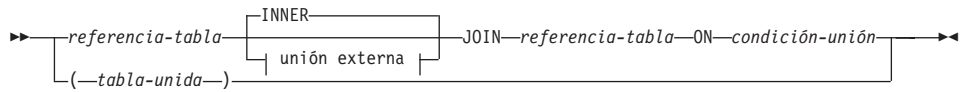
Pero los ejemplos siguientes no son válidos:

Ejemplo 4: **SELECT** t.c1, z.c5
FROM **TABLE**(tf6(t.c2)) **AS** z, t -- no puede resolver t en t.c2!
WHERE t.c3 = z.c4; -- compárese con el Ejemplo 1.

Ejemplo 5: **SELECT** a.c1, b.c5
FROM **TABLE**(tf7a(b.c2)) **AS** a, **TABLE**(tf7b(a.c6)) **AS** b
WHERE a.c3 = b.c4; -- no se puede resolver b en
 -- b.c2!

Ejemplo 6: **SELECT** d.deptno, d.deptname,
empinfo.avgsal, empinfo.empcount
FROM department d,
(SELECT **AVG**(e.salary) **AS** avgsal,
COUNT(*) **AS** empcount
FROM employee e -- department precede pero
WHERE e.workdept=d.deptno -- TABLE no se especifica
) AS empinfo; -- por tanto d.deptno es
 -- desconocido

tabla-unida



unión externa:



Una *tabla unida* especifica una tabla resultante intermedia que es el resultado de una unión interna o una unión externa. La tabla se deriva de aplicar uno de los operadores de unión: INNER, LEFT OUTER, RIGHT OUTER o FULL OUTER a sus operandos.

Se puede decir que las uniones internas son el producto cruzado de las tablas (combinación de cada fila de la tabla izquierda con cada fila de la tabla derecha), conservando sólo las filas en que la condición-unión es verdadera. Es posible que a la tabla resultante le falten filas de una o ambas tablas unidas. Las uniones externas incluyen la unión interna y conservan las filas que faltan. Hay tres tipos de uniones externas:

1. *unión externa izquierda* incluye las filas de la tabla de la izquierda que faltaban en la unión interna.
2. *unión externa derecha* incluye las filas de la tabla de la derecha que faltaban en la unión interna.
3. *unión externa completa* incluye las filas las tabla de la izquierda y de la derecha que faltaban en la unión interna.

Si no se especifica ningún operador-unión, el implícito es INNER. El orden en que se realizan múltiples uniones puede afectar al resultado. Las uniones pueden estar anidadas en otras uniones. El orden del proceso de uniones es generalmente de izquierda a derecha, pero se basa en la posición de la condición-unión necesaria. Es aconsejable utilizar paréntesis para que se pueda leer mejor el orden de las uniones anidadas. Por ejemplo:

```
tb1 left join tb2 on tb1.c1=tb2.c1
      right join tb3 left join tb4 on tb3.c1=tb4.c1
        on tb1.c1=tb3.c1
```

es igual que:

tabla-unida

```
(tb1 left join tb2 on tb1.c1=tb2.c1)
  right join (tb3 left join tb4 on tb3.c1=tb4.c1)
            on tb1.c1=tb3.c1
```

Una tabla unida se puede utilizar en cualquier contexto en el que se utilice cualquier forma de la sentencia SELECT. Una vista o un cursor es de sólo lectura si su sentencia SELECT incluye una tabla unida.

Una *condición-uniión* es una *condición-búsqueda* excepto en que:

- no puede contener ninguna subconsulta, escalar ni de cualquier otra clase
- no puede incluir ninguna operación de desreferencia ni una función Deref en que el valor de referencia sea diferente del de la columna de identificador de objeto.
- no puede incluir una función SQL
- cualquier columna a la que una expresión haga referencia de la *condición-uniión* debe ser una columna de una de las tablas de operandos de la unión asociada (en el ámbito de la misma cláusula tabla-unida)
- cualquier función a la que una expresión haga referencia de la *condición-uniión* de una unión externa completa debe ser determinante y no tener ninguna acción externa.

Se produce un error si la condición-uniión no cumple estas reglas (SQLSTATE 42972).

Las referencias a columnas se resuelven utilizando las reglas para la resolución de calificadores de nombres de columna. Las mismas reglas que se aplican a los predicados se aplican a las *condiciones-uniión* (consulte el apartado "Predicados" en la página 212).

Operaciones de unión

Una *condición-uniión* especifica emparejamientos de T1 y T2, donde T1 y T2 son tablas de los operandos izquierdo y derecho del operador JOIN de la *condición-uniión*. En todas las combinaciones posibles de filas de T1 y T2, una fila de T1 se empareja con una fila de T2 si la *condición-uniión* es verdadera. Cuando una fila de T1 se une con una fila de T2, una fila del resultado consta de los valores de dicha fila de T1 concatenada con los valores de dicha fila de T2. La ejecución puede implicar la generación de una fila nula. La fila nula de una tabla consta de un valor nulo para cada columna de la tabla, sin tener en cuenta si las columnas permiten valores nulos.

A continuación encontrará un resumen del resultado de las operaciones de unión:

- El resultado de T1 INNER JOIN T2 consta de sus filas emparejadas cuando la condición-uniión es verdadera.

- El resultado de T1 LEFT OUTER JOIN T2 consta de sus filas emparejadas cuando la condición-uni3n es verdadera y, para cada fila no emparejada de T1, la concatenaci3n de dicha fila con la fila nula de T2. Todas las columnas derivadas de T2 permiten valores nulos.
- El resultado de T1 RIGHT OUTER JOIN T2 consta de sus filas emparejadas cuando la condici3n-uni3n es verdadera y, para cada fila de T2 no emparejada, la concatenaci3n de dicha fila con la fila nula de T1. Todas las columnas derivadas de T1 permiten valores nulos.
- El resultado de T1 FULL OUTER JOIN T2 consta de sus filas emparejadas y, para cada fila de T2 no emparejada, la concatenaci3n de dicha fila con la fila nula de T1 y, para cada fila de T1 no emparejada, la concatenaci3n de dicha fila con la fila nula de T2. Todas las columnas derivadas de T1 y T2 permiten valores nulos.

cláusula-where

cláusula-where

►—WHERE—*condición-búsqueda*—◄

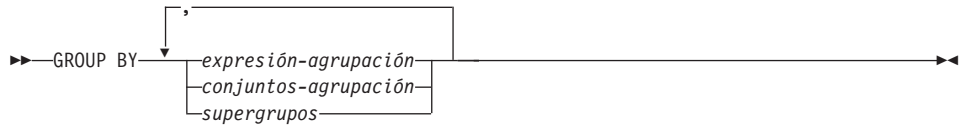
La cláusula WHERE especifica una tabla resultante intermedia que consta de aquellas filas de R para las que se cumple la *condición-búsqueda*. R es el resultado de la cláusula FROM de la subselección.

La *condición-búsqueda* debe ajustarse a las reglas siguientes:

- Cada *nombre-columna* debe identificar sin ambigüedades una columna de R o ser una referencia correlacionada. Un *nombre-columna* es una referencia correlacionada si identifica una columna de una *referencia-tabla* en una subselección externa.
- No debe especificarse una función de columna a menos que se especifique la cláusula WHERE en una subconsulta de una cláusula HAVING y el argumento de la función es una referencia correlacionada para un grupo.

Cualquier subconsulta de *condición-búsqueda* se ejecuta de forma efectiva para cada fila de R y los resultados se utilizan en la aplicación de la *condición-búsqueda* en la fila dada de R. Una subconsulta sólo se ejecuta en realidad para cada fila de R si incluye una referencia correlacionada. De hecho, una subconsulta sin referencias correlacionadas sólo se ejecuta una vez, mientras que una subconsulta con una referencia correlacionada puede tener que ejecutarse una vez para cada fila.

Cláusula group-by



La cláusula GROUP BY especifica una tabla resultante intermedia que consta de una agrupación de las filas de R. R es el resultado de la cláusula anterior de la subselección.

En su forma más simple, una cláusula GROUP BY contiene una *expresión de agrupación*. Una expresión de agrupación es una *expresión* que se utiliza para definir la agrupación de R. Cada *nombre-columna* incluido en la expresión-agrupación debe identificar de forma inequívoca una columna de R (SQLSTATE 42702 o 42703). El atributo de longitud de cada expresión de agrupación no debe tener más de 255 bytes (SQLSTATE 42907). Una expresión de agrupación no puede incluir una selección completa escalar (SQLSTATE 42822) ni ninguna función que sea una variante o que tenga una acción externa (SQLSTATE 42845).

Las formas más complejas de la cláusula GROUP BY son los *conjuntos-agrupación* y los *supergrupos*. Para ver una descripción de estas formas, consulte los apartados “conjuntos-agrupación” en la página 488 y “supergrupos” en la página 489, respectivamente.

El resultado de GROUP BY es un conjunto de grupos de filas. Cada fila del resultado representa el conjunto de filas para el que la *expresión-agrupación* es igual. En la agrupación, todos los valores nulos de una *expresión-agrupación* se consideran iguales.

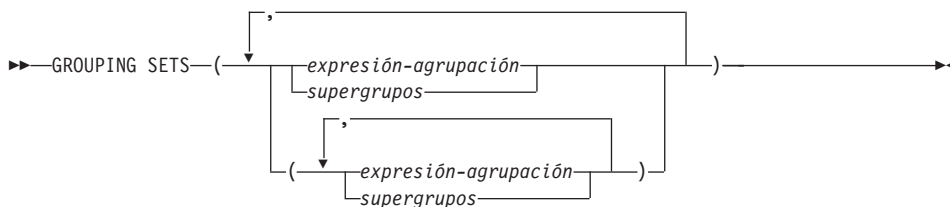
Una *expresión-agrupación* se puede utilizar en una condición de búsqueda de una cláusula HAVING, en una expresión de una cláusula SELECT o en una *expresión-clave-clasificación* de una cláusula ORDER BY (consulte el apartado “cláusula-order-by” en la página 522 para ver los detalles). En cada caso, la referencia sólo especifica un valor para cada grupo. Por ejemplo, si la *expresión-agrupación* es $col1+col2$, una expresión permitida en la lista de selección sería $col1+col2+3$. Las reglas de asociación para expresiones rechazarían la expresión parecida $3+col1+col2$, a menos que se utilicen paréntesis para asegurar que la expresión correspondiente se evalúe en el mismo orden. Por lo tanto, $3+(col1+col2)$ también se permitiría en la lista de selección. Si se utiliza el operador de concatenación, la *expresión-agrupación* debe utilizarse exactamente como se ha especificado la expresión en la lista de selección.

cláusula-group-by

Si la *expresión-agrupación* contiene series de longitud variable con blancos de cola, los valores del grupo pueden diferir en el número de blancos de cola y puede que no todos tengan la misma longitud. En dicho caso, la referencia a la *expresión-agrupación* continúa especificando sólo un valor para cada grupo, pero el valor para un grupo se elige arbitrariamente entre el conjunto de valores disponibles. Por lo tanto, la longitud real del valor del resultado es imprevisible.

Tal como se ha apuntado, existen casos en los que la cláusula GROUP BY no puede hacer referencia directamente a una columna que esté especificada en la cláusula SELECT como una expresión (selección completa-escalar, variante o funciones de acción externa). Para agrupar utilizando una expresión como ésta, utilice una expresión de tabla anidada o una expresión de tabla común para proporcionar primero una tabla resultante con la expresión como una columna del resultado. Para ver un ejemplo de la utilización de expresiones de tabla anidadas, consulte el “Ejemplo A9” en la página 497.

conjuntos-agrupación



Una especificación de *conjuntos-agrupación* permite especificar múltiples cláusulas de agrupación en una sola sentencia. Se puede decir que es la unión de dos o más grupos de filas en un solo conjunto resultante. Es lógicamente equivalente a la unión de múltiples subselecciones con la cláusula group by en cada subselección correspondiente a un conjunto de agrupación. Un conjunto de agrupación puede ser un solo elemento o puede ser una lista de elementos delimitados por paréntesis, donde un elemento es una *expresión-agrupación* o un *supergrupo*. La utilización de *conjuntos-agrupación* permite calcular los grupos con una sola pasada por la tabla base.

La especificación de *conjuntos-agrupación* permite utilizar una *expresión-agrupación* simple o las formas más complejas de *supergrupos*. Para ver una descripción de *supergrupos*, consulte el apartado “supergrupos” en la página 489.

Tenga en cuenta que los conjuntos de agrupación son el bloque fundamental para la construcción de operaciones GROUP BY. Una operación group by simple con una sola columna puede considerarse un conjunto de agrupación con un elemento. Por ejemplo:

GROUP BY

es igual a

GROUP BY GROUPING SET((a))

y

GROUP BY a,b,c

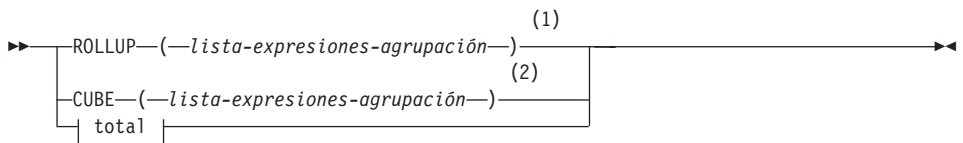
es igual a

GROUP BY GROUPING SET((a,b,c))

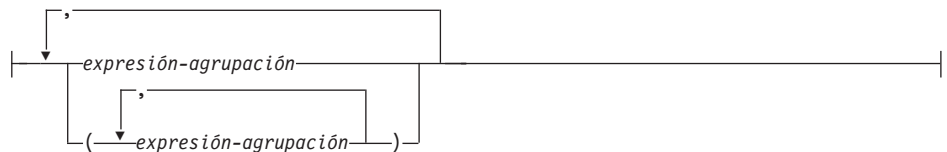
Las columnas de no agregación de la lista de selección de la subselección que se excluyen de un conjunto de agrupación devolverán un nulo para dichas columnas para cada fila generada para dicho conjunto de agrupación. Esto refleja el hecho que la agregación se ha realizado sin tener en cuenta los valores para dichas columnas. Consulte el apartado "GROUPING" en la página 266 para ver la forma de distinguir las filas con nullos de los datos reales de las filas con nullos generadas por conjuntos de agrupación.

Desde el "Ejemplo C2" en la página 504 al "Ejemplo C7" en la página 508 se ilustra la utilización de los conjuntos de agrupación.

supergrupos



lista-expresiones-agrupación:



cláusula-group-by

total:

|—(—)|

Notas:

- 1 Una especificación alternativa cuando se utiliza sola en la cláusula Group By es: lista-expresiones-agrupación WITH ROLLUP.
- 2 Una especificación alternativa cuando se utiliza sola en la cláusula Group By es: lista-expresiones-agrupación WITH CUBE.

ROLLUP (lista-expresiones-agrupación)

Una *agrupación ROLLUP* es una extensión de la cláusula GROUP BY que produce un conjunto resultante que contiene filas de *subtotales* además de las filas agrupadas "normales". Las filas de *subtotales*⁵³ son filas "superagregadas" que contienen más agregados cuyos valores se obtienen al aplicar las mismas funciones de columna que se han utilizado para obtener las filas agrupadas.

Una agrupación ROLLUP es una serie de *conjuntos-agrupación*. La especificación general de ROLLUP con n elementos

GROUP BY ROLLUP($C_1, C_2, \dots, C_{n-1}, C_n$)

es equivalente a

GROUP BY GROUPING SETS(($C_1, C_2, \dots, C_{n-1}, C_n$)
(C_1, C_2, \dots, C_{n-1})

⋮
(C_1, C_2)
(C_1)
()

Observe que los n elementos de ROLLUP se convierten en $n+1$ conjuntos de agrupación.

Tenga en cuenta que el orden en el que se especifican las *expresiones-agrupación* es significativo para ROLLUP. Por ejemplo:

GROUP BY ROLLUP(a, b)

es equivalente a

GROUP BY GROUPING SETS((a, b)
(a)
()

53. Se llaman filas de subtotales porque es su utilización más normal, sin embargo, puede utilizarse cualquier función de columna para la agregación. Por ejemplo, MAX y AVG se utilizan en "Ejemplo C8" en la página 510.

mientras que

GROUP BY ROLLUP(b, a)

es igual a

GROUP BY GROUPING SETS((b, a)
(b)
())

La cláusula ORDER BY es la única manera de garantizar el orden de las filas en el conjunto resultante. El “Ejemplo C3” en la página 504 ilustra la utilización de ROLLUP.

CUBE (*lista-expresiones-agrupación*)

Una *agrupación CUBE* es una extensión a la cláusula GROUP BY que produce un conjunto resultante que contiene todas las filas de la agrupación ROLLUP y, además, contiene filas de “tabulación cruzada”. Las filas de *tabulación cruzada* son filas “superagregadas” adicionales que no forman parte de una agrupación con subtotales.

Igual que ROLLUP, una agrupación CUBE también puede decirse que es una serie de *conjuntos-agrupación*. En el caso de CUBE, todas las permutaciones de la *lista-expresiones-agregación* al cubo se calcula junto con el total. Por lo tanto, los n elementos de CUBE se convierten en 2^{*n} (2 elevado a la potencia n) *conjuntos-agrupación*. Por ejemplo, una especificación de

GROUP BY CUBE(a, b, c)

es equivalente a

GROUP BY GROUPING SETS((a, b, c)
(a, b)
(a, c)
(b, c)
(a)
(b)
(c)
())

Observe que los 3 elementos de CUBE se convierten en 8 conjuntos de agrupaciones.

El orden de especificación de los elementos no importa para CUBE. ‘CUBE (DayOfYear, Sales_Person)’ y ‘CUBE (Sales_Person, DayOfYear)’ dan los mismos conjuntos del resultado. La utilización de la palabra ‘mismos’ se aplica al contenido del conjunto resultante, no a su orden. La cláusula ORDER BY es la única manera de garantizar el orden de las filas en el conjunto resultante. El “Ejemplo C4” en la página 505 ilustra la utilización de CUBE.

cláusula-group-by

lista-expresiones-agrupación

Una *lista-expresiones-agrupación* se utiliza en la cláusula CUBE o ROLLUP para definir el número de elementos de la operación CUBE o ROLLUP. Se controla utilizando los paréntesis para delimitar los elementos con múltiples *expresiones-agrupación*.

Las reglas para la *expresión-agrupación* se describen en el apartado “Cláusula group-by” en la página 487. Por ejemplo, suponga que una consulta tiene que devolver los gastos totales para ROLLUP de City dentro de una Province pero no de un County. Sin embargo la cláusula:

```
GROUP BY ROLLUP(Province, County, City)
```

da como resultado filas de subtotales que no se desean para County. En la cláusula

```
GROUP BY ROLLUP(Province, (County, City))
```

el compuesto (County, City) forma un elemento de ROLLUP y, por lo tanto, una consulta que utiliza esta cláusula dará el resultado deseado. En otras palabras, ROLLUP de dos elementos

```
GROUP BY ROLLUP(Province, (County, City))
```

genera

```
GROUP BY GROUPING SETS((Province, County, City)  
                          (Province)  
                          () )
```

mientras que ROLLUP de 3 elementos generaría

```
GROUP BY GROUPING SETS((Province, County, City)  
                          (Province, County)  
                          (Province)  
                          () )
```

El “Ejemplo C2” en la página 504 también utiliza valores de columna compuestos.

total

Tanto CUBE como ROLLUP devuelven una fila que es la agregación global (total). Esto se puede especificar por separado con paréntesis vacíos dentro de la cláusula GROUPING SET. También puede especificarse directamente en la cláusula GROUP BY, aunque no surte ningún efecto en el resultado de la consulta. El “Ejemplo C4” en la página 505 utiliza la sintaxis del total.

Combinación de conjuntos de agrupación

Se puede utilizar para combinar cualquier tipo de cláusula GROUP BY. Cuando se combinan los campos de una *expresión-agrupación* simple con otros grupos, se “añaden” al principio de los *conjuntos de agrupación* resultantes.

Cuando se combinan las expresiones *ROLLUP* o *CUBE*, funcionan como "multiplicadores" en el resto de la expresión, formando entradas de un conjunto de agrupación adicional según la definición de *ROLLUP* o *CUBE*.

Por ejemplo, la combinación de elementos de *expresión-agrupación* actúa de la siguiente manera:

GROUP BY a, ROLLUP(b,c)

es equivalente a

**GROUP BY GROUPING SETS((a,b,c)
(a,b)
(a))**

O de manera parecida,

GROUP BY a, b, ROLLUP(c,d)

es equivalente a

**GROUP BY GROUPING SETS((a,b,c,d)
(a,b,c)
(a,b))**

La combinación de elementos de *ROLLUP* actúa de la siguiente manera:

GROUP BY ROLLUP(a), ROLLUP(b,c)

es equivalente a

**GROUP BY GROUPING SETS((a,b,c)
(a,b)
(a)
(b,c)
(b)
())**

De manera similar,

GROUP BY ROLLUP(a), CUBE(b,c)

es equivalente a

**GROUP BY GROUPING SETS((a,b,c)
(a,b)
(a,c)
(a)
(b,c)
(b)
(c)
())**

La combinación de elementos de *CUBE* y de *ROLLUP* actúa de la siguiente manera:

cláusula-group-by

```
GROUP BY CUBE(a,b), ROLLUP(c,d)
```

es equivalente a

```
GROUP BY GROUPING SETS((a,b,c,d)  
  (a,b,c)  
  (a,b)  
  (a,c,d)  
  (a,c)  
  (a)  
  (b,c,d)  
  (b,c)  
  (b)  
  (c,d)  
  (c)  
  ( ) )
```

Igual que una *expresión-agrupación* simple, la combinación de conjuntos de agrupación elimina también los duplicados dentro de cada conjunto de agrupación. Por ejemplo,

```
GROUP BY a, ROLLUP(a,b)
```

es equivalente a

```
GROUP BY GROUPING SETS((a,b)  
  (a) )
```

Un ejemplo más completo de la combinación de conjuntos de agrupación es crear un conjunto resultante que elimine ciertas filas que se devolverían para una agregación CUBE completa.

Por ejemplo, considere la siguiente cláusula GROUP BY:

```
GROUP BY Region,  
  ROLLUP(Sales_Person, WEEK(Sales_Date)),  
  CUBE(YEAR(Sales_Date), MONTH (Sales_Date))
```

La columna listada inmediatamente a la derecha de GROUP BY está agrupada simplemente, las que están entre paréntesis a continuación de ROLLUP se han avanzado y las que están entre paréntesis a continuación de CUBE se han elevado al cubo. Por lo tanto, la cláusula anterior da como resultado el cubo de MONTH en YEAR que después avanza en WEEK en Sales_Person en la agregación Region. No da como resultado una fila del total ni ninguna fila de tabulación cruzada en Región, Sales_Person o WEEK(Sales_Date) por lo que produce menos filas que las de la cláusula:

```
GROUP BY ROLLUP (Region, Sales_Person, WEEK(Sales_Date),  
  YEAR(Sales_Date), MONTH(Sales_Date) )
```

cláusula-having

La cláusula HAVING especifica una tabla resultante intermedia que consta de aquellos grupos de R para los que la *condición-búsqueda* es verdadera. R es el resultado de la cláusula anterior de la subselección. Si esta cláusula no es GROUP BY, R se considera un solo grupo sin columnas de agrupación.

Cada *nombre-columna* de la condición de búsqueda debe realizar una de las acciones siguientes:

- Identificar sin ambigüedades una columna de agrupación de R.
- Estar especificado dentro de la función de columna.
- Ser una referencia correlacionada. Un *nombre-columna* es una referencia correlacionada si identifica una columna de una *referencia-tabla* en una subselección externa.

Un grupo de R al que se aplica la condición de búsqueda suministra el argumento para cada función de columna en la condición de búsqueda, excepto para cualquier función cuyo argumento sea una referencia correlacionada.

Si la condición de búsqueda contiene una subconsulta, puede considerarse que la subconsulta se ejecuta cada vez que se aplica la condición de búsqueda a un grupo de R y los resultados se utilizan en la aplicación de la condición de búsqueda. En realidad, la subconsulta sólo se ejecuta para cada grupo si contiene una referencia correlacionada. Para ver una ilustración de la diferencia, consulte el “Ejemplo A6” en la página 496 y el “Ejemplo A7” en la página 497.

Una referencia correlacionada a un grupo de R debe identificar una columna de agrupación o estar contenida en una función de columna.

Cuando se utiliza HAVING sin GROUP BY, la lista de selección sólo puede ser un nombre de columna dentro de una función de columna, una referencia correlacionada a columna, un literal o un registro especial.

Ejemplos de subselecciones

Ejemplos de subselecciones

Ejemplo A1: Seleccione todas las columnas y filas de la tabla EMPLOYEE.

```
SELECT * FROM EMPLOYEE
```

Ejemplo A2: Una las tablas EMP_ACT y EMPLOYEE, seleccione todas las columnas de la tabla EMP_ACT y añada el apellido del empleado (LASTNAME) de la tabla EMPLOYEE a cada fila del resultado.

```
SELECT EMP_ACT.*, LASTNAME  
FROM EMP_ACT, EMPLOYEE  
WHERE EMP_ACT.EMPNO = EMPLOYEE.EMPNO
```

Ejemplo A3: Una las tablas EMPLOYEE y DEPARTMENT, seleccione el número del empleado (EMPNO), el apellido del empleado (LASTNAME), el número del departamento (WORKDEPT en la tabla EMPLOYEE y DEPTNO en la tabla DEPARTMENT) y el nombre del departamento (DEPTNAME) de todos los empleados que han nacido (BIRTHDATE) con anterioridad a 1930.

```
SELECT EMPNO, LASTNAME, WORKDEPT, DEPTNAME  
FROM EMPLOYEE, DEPARTMENT  
WHERE WORKDEPT = DEPTNO  
AND YEAR(BIRTHDATE) < 1930
```

Ejemplo A4: Seleccione el trabajo (JOB) y los salarios máximo y mínimo (SALARY) de cada grupo de filas con el mismo código de trabajo en la tabla EMPLOYEE, pero sólo para los grupos con más de una fila y con un salario máximo mayor o igual que 27000.

```
SELECT JOB, MIN(SALARY), MAX(SALARY)  
FROM EMPLOYEE  
GROUP BY JOB  
HAVING COUNT(*) > 1  
AND MAX(SALARY) >= 27000
```

Ejemplo A5: Seleccione todas las filas de la tabla EMP_ACT para los empleados (EMPNO) del departamento (WORKDEPT) 'E11'. (Los números del departamento del empleado se muestran en la tabla EMPLOYEE.)

```
SELECT *  
FROM EMP_ACT  
WHERE EMPNO IN  
(SELECT EMPNO  
FROM EMPLOYEE  
WHERE WORKDEPT = 'E11')
```

Ejemplo A6: En la tabla EMPLOYEE, seleccione el número de departamento (WORKDEPT) y el salario (SALARY) máximo del departamento para todos los departamentos cuyo salario máximo sea menor que el salario promedio de todos los empleados.

```

SELECT WORKDEPT, MAX(SALARY)
      FROM EMPLOYEE
   GROUP BY WORKDEPT
  HAVING MAX(SALARY) < (SELECT AVG(SALARY)
                        FROM EMPLOYEE)

```

La subconsulta de la cláusula HAVING sólo se ejecutará una vez en este ejemplo.

Ejemplo A7: Utilizando la tabla EMPLOYEE, seleccione el número de departamento (WORKDEPT) y el salario (SALARY) máximo del departamento para todos los departamentos cuyo salario máximo sea menor que el salario promedio de los demás departamentos.

```

SELECT WORKDEPT, MAX(SALARY)
      FROM EMPLOYEE EMP_COR
   GROUP BY WORKDEPT
  HAVING MAX(SALARY) < (SELECT AVG(SALARY)
                        FROM EMPLOYEE
                       WHERE NOT WORKDEPT = EMP_COR.WORKDEPT)

```

A diferencia del “Ejemplo A6” en la página 496, la subconsulta de la cláusula HAVING se habrá de ejecutar para cada grupo.

Ejemplo A8: Determine el número de empleado y el salario de los representantes de ventas junto con el salario promedio y cuenta punta de sus departamentos.

Esta consulta primero debe crear una expresión de tabla anidada (DINFO) para obtener las columnas AVGSALARY y EMPCOUNT, así como la columna DEPTNO que se utiliza en la cláusula WHERE.

```

SELECT THIS_EMP.EMPNO, THIS_EMP.SALARY, DINFO.AVGSALARY, DINFO.EMPCOUNT
      FROM EMPLOYEE THIS_EMP,
           (SELECT OTHERS.WORKDEPT AS DEPTNO,
              AVG(OTHERS.SALARY) AS AVGSALARY,
              COUNT(*) AS EMPCOUNT
            FROM EMPLOYEE OTHERS
           GROUP BY OTHERS.WORKDEPT
           ) AS DINFO
  WHERE THIS_EMP.JOB = 'SALESREP'
     AND THIS_EMP.WORKDEPT = DINFO.DEPTNO

```

Utilizar una expresión de tabla anidada para este caso ahorra la actividad general de crear la vista DIFO como una vista normal. Durante la preparación de la sentencia, se evita el acceso al catálogo para la vista y, debido al contexto del resto de la consulta, sólo se han de tener en cuenta las filas para el departamento de representantes de ventas para la vista.

Ejemplo A9: Visualice el nivel de formación promedio y el salario de 5 grupos de empleados al azar.

Ejemplos de subselecciones

Esta consulta necesita la utilización de una expresión de tabla anidada para establecer el valor aleatorio de cada empleado para que pueda utilizarse posteriormente en la cláusula GROUP BY.

```
SELECT RANDID , AVG(EDLEVEL), AVG(SALARY)
  FROM ( SELECT EDLEVEL, SALARY, INTEGER(RAND()*5) AS RANDID
        FROM EMPLOYEE
        ) AS EMPRAND
  GROUP BY RANDID
```


Ejemplos de uniones

Ejemplo B1: Este ejemplo ilustra el resultado de varias uniones utilizando las tablas J1 y J2. Estas tablas contienen las filas que se muestran.

```
SELECT * FROM J1
```

W	X
A	11
B	12
C	13

```
SELECT * FROM J2
```

Y	Z
A	21
C	22
D	23

La siguiente consulta realiza una unión interna de J1 y J2, emparejando la primera columna de ambas tablas.

```
SELECT * FROM J1 INNER JOIN J2 ON W=Y
```

W	X	Y	Z
A	11	A	21
C	13	C	22

En este ejemplo de unión interna, la fila con la columna W='C' de J1 y la fila con la columna Y='D' de J2 no se incluyen en el resultado porque no tienen una coincidencia en la otra tabla. Observe que la forma alternativa siguiente de una consulta de unión interna genera el mismo resultado.

```
SELECT * FROM J1, J2 WHERE W=Y
```

La unión externa izquierda siguiente recuperará la fila que falta de J1 con nulos para las columnas de J2. Se incluyen todas las filas de J1.

```
SELECT * FROM J1 LEFT OUTER JOIN J2 ON W=Y
```

W	X	Y	Z
A	11	A	21
B	12	-	-
C	13	C	22

La siguiente unión externa derecha recuperará la fila que falta de J2 con nulos para las columnas de J1. Se incluyen todas las filas de J2.

Ejemplos de uniones

```
SELECT * FROM J1 RIGHT OUTER JOIN J2 ON W=Y
```

W	X	Y	Z
A	11	A	21
C	13	C	22
-	-	D	23

La siguiente unión externa completa recuperará las filas que faltan de las dos tablas J1 y J2, con nulos cuando sea adecuado. Se incluyen todas las filas de las tablas J1 y J2.

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y
```

W	X	Y	Z
A	11	A	21
C	13	C	22
-	-	D	23
B	12	-	-

Ejemplo B2: Utilizando las tablas J1 y J2 del ejemplo anterior, examine lo que pasa cuando se añade un predicado adicional a la condición de búsqueda.

```
SELECT * FROM J1 INNER JOIN J2 ON W=Y AND X=13
```

W	X	Y	Z
C	13	C	22

La condición adicional ha provocado que la unión interna sólo seleccionase 1 fila en comparación con la unión interna del “Ejemplo B1” en la página 499.

Observe el impacto que tiene en la unión externa completa.

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y AND X=13
```

W	X	Y	Z
-	-	A	21
C	13	C	22
-	-	D	23
A	11	-	-
B	12	-	-

Ahora el resultado tiene 5 filas (a diferencia de 4 sin el predicado adicional) ya que sólo había 1 fila en la unión interna y tienen que devolverse todas las filas de ambas tablas.

La siguiente consulta ilustra que la colocación del mismo predicado adicional en la cláusula WHERE provoca resultados completamente diferentes.

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y
WHERE X=13
```

W	X	Y	Z
---	-----	---	-----
C	13	C	22

La cláusula WHERE se aplica después del resultado intermedio de la unión externa completa. Este resultado intermedio sería el mismo que el resultado de la consulta de unión externa completa del “Ejemplo B1” en la página 499. La cláusula WHERE se aplica a este resultado intermedio y elimina todas las filas excepto la que contiene X=13. La elección de la ubicación de un predicado cuando se realizan uniones externas puede afectar significativamente a los resultados. Examine lo que pasa si el predicado es X=12 en lugar de X=13. La siguiente unión interna no devuelve ninguna fila.

```
SELECT * FROM J1 INNER JOIN J2 ON W=Y AND X=12
```

Por lo tanto, la unión externa completa devolvería 6 filas: 3 filas de J1 con nulos para las columnas de J2, y 3 filas de J2 con nulos para las columnas de J1.

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y AND X=12
```

W	X	Y	Z
---	-----	---	-----
-		- A	21
-		- C	22
-		- D	23
A	11	-	-
B	12	-	-
C	13	-	-

En cambio, si el predicado adicional está en la cláusula WHERE, se devuelve la fila 1.

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y
WHERE X=12
```

W	X	Y	Z
---	-----	---	-----
B	12	-	-

Ejemplo B3: Liste todos los departamentos con el número de empleado y el apellido del director, incluyendo los departamentos sin director.

```
SELECT DEPTNO, DEPTNAME, EMPNO, LASTNAME
FROM DEPARTMENT LEFT OUTER JOIN EMPLOYEE
ON MGRNO = EMPNO
```

Ejemplo B4: Liste todos los números de empleado y el apellido con el número de empleado y el apellido de su director, incluyendo los empleados sin director.

Ejemplos de uniones

```
SELECT E.EMPNO, E.LASTNAME, M.EMPNO, M.LASTNAME
FROM EMPLOYEE E LEFT OUTER JOIN
                                DEPARTMENT INNER JOIN EMPLOYEE M
ON MGRNO = M.EMPNO
ON E.WORKDEPT = DEPTNO
```

La unión interna determina el apellido de cualquier director identificado en la tabla DEPARTMENT y la unión externa izquierda garantiza que se listen todos los empleados incluso si no se encuentra un departamento correspondiente en DEPARTMENT.

Ejemplos de conjuntos de agrupación, Cube y Rollup

Las consultas del “Ejemplo C1” al “Ejemplo C4” en la página 505 utilizan un subconjunto de filas de las tablas SALES basadas en el predicado ‘WEEK(SALES_DATE) = 13’.

```
SELECT WEEK(SALES_DATE) AS WEEK,
        DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
        SALES_PERSON, SALES AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
```

lo que da como resultado:

WEEK	DAY_WEEK	SALES_PERSON	UNITS_SOLD
13	6	LUCCHESSI	3
13	6	LUCCHESSI	1
13	6	LEE	2
13	6	LEE	2
13	6	LEE	3
13	6	LEE	5
13	6	GOUNOT	3
13	6	GOUNOT	1
13	6	GOUNOT	7
13	7	LUCCHESSI	1
13	7	LUCCHESSI	2
13	7	LUCCHESSI	1
13	7	LEE	7
13	7	LEE	3
13	7	LEE	7
13	7	LEE	4
13	7	GOUNOT	2
13	7	GOUNOT	18
13	7	GOUNOT	1

Ejemplo C1: Esta es una consulta con una cláusula GROUP BY básica en 3 columnas:

```
SELECT WEEK(SALES_DATE) AS WEEK,
        DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
        SALES_PERSON, SUM(SALES) AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
GROUP BY WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE), SALES_PERSON
ORDER BY WEEK, DAY_WEEK, SALES_PERSON
```

Da como resultado:

WEEK	DAY_WEEK	SALES_PERSON	UNITS_SOLD
13	6	GOUNOT	11
13	6	LEE	12

Ejemplos de conjuntos de agrupación, Cube y Rollup

13	6 LUCCHESSI	4
13	7 GOUNOT	21
13	7 LEE	21
13	7 LUCCHESSI	4

Ejemplo C2: Genere el resultado basándose en dos conjuntos de agrupación diferentes de las filas de la tabla SALES.

```
SELECT WEEK(SALES_DATE) AS WEEK,
        DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
        SALES_PERSON, SUM(SALES) AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
GROUP BY GROUPING SETS ( (WEEK(SALES_DATE), SALES_PERSON),
                          (DAYOFWEEK(SALES_DATE), SALES_PERSON))
ORDER BY WEEK, DAY_WEEK, SALES_PERSON
```

Esto da como resultado:

WEEK	DAY_WEEK	SALES_PERSON	UNITS_SOLD
-----	-----	-----	-----
13	-	GOUNOT	32
13	-	LEE	33
13	-	LUCCHESSI	8
-	6	GOUNOT	11
-	6	LEE	12
-	6	LUCCHESSI	4
-	7	GOUNOT	21
-	7	LEE	21
-	7	LUCCHESSI	4

Las filas con WEEK 13 son del primer conjunto de agrupación y las demás filas son del segundo conjunto de agrupación.

Ejemplo C3: Si utiliza las 3 columnas diferenciadas implicadas en los conjuntos de agrupación del “Ejemplo C2” y lleva a cabo ROLLUP, puede ver los conjuntos de agrupación para (WEEK,DAY_WEEK,SALES_PERSON), (WEEK, DAY_WEEK), (WEEK) y el total.

```
SELECT WEEK(SALES_DATE) AS WEEK,
        DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
        SALES_PERSON, SUM(SALES) AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
GROUP BY ROLLUP ( WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE), SALES_PERSON )
ORDER BY WEEK, DAY_WEEK, SALES_PERSON
```

Esto da como resultado:

WEEK	DAY_WEEK	SALES_PERSON	UNITS_SOLD
-----	-----	-----	-----
13	6	GOUNOT	11
13	6	LEE	12
13	6	LUCCHESSI	4
13	6	-	27

Ejemplos de conjuntos de agrupación, Cube y Rollup

13	7 GOUNOT	21
13	7 LEE	21
13	7 LUCCHESSI	4
13	7 -	46
13	- -	73
-	- -	73

Ejemplo C4: Si ejecuta la misma consulta que el “Ejemplo C3” en la página 504 sustituyendo sólo ROLLUP por CUBE, podrá ver conjuntos de agrupación adicionales para (WEEK,SALES_PERSON), (DAY_WEEK,SALES_PERSON), (DAY_WEEK), (SALES_PERSON) en el resultado.

```

SELECT WEEK(SALES_DATE) AS WEEK,
                                DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
                                SALES_PERSON, SUM(SALES) AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
GROUP BY CUBE ( WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE), SALES_PERSON )
ORDER BY WEEK, DAY_WEEK, SALES_PERSON
    
```

Da como resultado:

WEEK	DAY_WEEK	SALES_PERSON	UNITS_SOLD
-----	-----	-----	-----
13	6	GOUNOT	11
13	6	LEE	12
13	6	LUCCHESSI	4
13	6	-	27
13	7	GOUNOT	21
13	7	LEE	21
13	7	LUCCHESSI	4
13	7	-	46
13	-	GOUNOT	32
13	-	LEE	33
13	-	LUCCHESSI	8
13	-	-	73
-	6	GOUNOT	11
-	6	LEE	12
-	6	LUCCHESSI	4
-	6	-	27
-	7	GOUNOT	21
-	7	LEE	21
-	7	LUCCHESSI	4
-	7	-	46
-	-	GOUNOT	32
-	-	LEE	33
-	-	LUCCHESSI	8
-	-	-	73

Ejemplo C5: Obtenga un conjunto resultante que incluya un total de filas seleccionadas de la tabla SALES junto con un grupo de filas agregadas por SALES_PERSON y MONTH.

Ejemplos de conjuntos de agrupación, Cube y Rollup

```
SELECT SALES_PERSON,  
       MONTH(SALES_DATE) AS MONTH,  
       SUM(SALES) AS UNITS_SOLD  
FROM SALES  
GROUP BY GROUPING SETS ( (SALES_PERSON, MONTH(SALES_DATE)),  
                          ()  
                        )  
ORDER BY SALES_PERSON, MONTH
```

Esto da como resultado:

SALES_PERSON	MONTH	UNITS_SOLD
GOUNOT	3	35
GOUNOT	4	14
GOUNOT	12	1
LEE	3	60
LEE	4	25
LEE	12	6
LUCCHESSI	3	9
LUCCHESSI	4	4
LUCCHESSI	12	1
- -		155

Ejemplo C6: Este ejemplo muestra dos consultas ROLLUP simples seguidas de una consulta que trata los dos ROLLUP como conjuntos de agrupación en un sólo conjunto resultante y especifica el orden de filas para cada columna implicada en los conjuntos de agrupación.

Ejemplo C6-1:

```
SELECT WEEK(SALES_DATE) AS WEEK,  
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,  
       SUM(SALES) AS UNITS_SOLD  
FROM SALES  
GROUP BY ROLLUP ( WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE) )  
ORDER BY WEEK, DAY_WEEK
```

da como resultado:

WEEK	DAY_WEEK	UNITS_SOLD
13	6	27
13	7	46
13	-	73
14	1	31
14	2	43
14	-	74
53	1	8
53	-	8
- -		155

Ejemplo C6-2:

Ejemplos de conjuntos de agrupación, Cube y Rollup

```

SELECT MONTH(SALES_DATE) AS MONTH,
       REGION,
       SUM(SALES) AS UNITS_SOLD
FROM SALES
GROUP BY ROLLUP ( MONTH(SALES_DATE), REGION );
ORDER BY MONTH, REGION

```

da como resultado:

MONTH	REGION	UNITS_SOLD
-----	-----	-----
3	Manitoba	22
3	Ontario-North	8
3	Ontario-South	34
3	Quebec	40
3	-	104
4	Manitoba	17
4	Ontario-North	1
4	Ontario-South	14
4	Quebec	11
4	-	43
12	Manitoba	2
12	Ontario-South	4
12	Quebec	2
12	-	8
-	-	155

Ejemplo C6-3:

```

SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       MONTH(SALES_DATE) AS MONTH,
       REGION,
       SUM(SALES) AS UNITS_SOLD
FROM SALES
GROUP BY GROUPING SETS ( ROLLUP( WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE) ),
                        ROLLUP( MONTH(SALES_DATE), REGION ) )
ORDER BY WEEK, DAY_WEEK, MONTH, REGION

```

da como resultado:

WEEK	DAY_WEEK	MONTH	REGION	UNITS_SOLD
-----	-----	-----	-----	-----
13	6	-	-	27
13	7	-	-	46
13	-	-	-	73
14	1	-	-	31
14	2	-	-	43
14	-	-	-	74
53	1	-	-	8
53	-	-	-	8
-	-	3	Manitoba	22
-	-	3	Ontario-North	8
-	-	3	Ontario-South	34
-	-	3	Quebec	40
-	-	3	-	104

Ejemplos de conjuntos de agrupación, Cube y Rollup

-	-	4 Manitoba	17
-	-	4 Ontario-North	1
-	-	4 Ontario-South	14
-	-	4 Quebec	11
-	-	4 -	43
-	-	12 Manitoba	2
-	-	12 Ontario-South	4
-	-	12 Quebec	2
-	-	12 -	8
-	-	- -	155
-	-	- -	155

La utilización de los dos ROLLUP como conjuntos de agrupación hace que el resultado incluya filas duplicadas. Incluso hay dos filas del total.

Observe cómo la utilización de ORDER BY ha afectado al resultado:

- En el primer conjunto agrupado, la semana 53 se ha cambiado a la posición final.
- En el segundo conjunto agrupado, el mes 12 se ha puesto al final y las regiones aparecen por orden alfabético.
- Los valores nulos se clasifican arriba.

Ejemplo C7: En las consultas que realizan varios ROLLUP en una sola pasada (como por ejemplo, "Ejemplo C6-3" en la página 507) tiene la posibilidad de indicar, si lo desea, qué conjunto de agrupación ha producido cada fila. Los pasos siguientes demuestran cómo proporcionar una columna (denominada GROUP) que indica el origen de cada fila del conjunto resultante. Por origen se quiere decir cual de los dos conjuntos de agrupación ha producido la fila del conjunto resultante.

Paso 1: Introduzca una manera de "generar" los nuevos valores de datos, utilizando una consulta que los selecciona en la cláusula VALUES (que es una forma alternativa de una selección completa). Esta consulta muestra cómo se puede derivar una tabla llamada "X" que tienen 2 columnas "R1" y "R2" y 1 fila de datos.

```
SELECT R1,R2
FROM (VALUES('GROUP 1','GROUP 2')) AS X(R1,R2);
```

da como resultado:

```
  R1      R2
-----
GROUP 1  GROUP 2
```

Paso 2: Forme el producto cruzado de esta tabla "X" con la tabla SALES. Esto añade las columnas "R1" y "R2" a cada fila.

Ejemplos de conjuntos de agrupación, Cube y Rollup

```

SELECT R1, R2, WEEK(SALES_DATE) AS WEEK,
        DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
        MONTH(SALES_DATE) AS MONTH,
        REGION,
        SALES AS UNITS_SOLD
FROM SALES, (VALUES('GROUP 1', 'GROUP 2')) AS X(R1,R2)

```

Esto añade las columnas "R1" y "R2" a cada fila.

Paso 3: Ahora se pueden combinar estas columnas con los conjuntos de agrupación para que incluyan estas columnas en el análisis de avance.

```

SELECT R1, R2,
        WEEK(SALES_DATE) AS WEEK,
        DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
        MONTH(SALES_DATE) AS MONTH,
        REGION, SUM(SALES) AS UNITS_SOLD
FROM SALES, (VALUES('GROUP 1', 'GROUP 2')) AS X(R1,R2)
GROUP BY GROUPING SETS ((R1, ROLLUP(WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE))),
(R2, ROLLUP( MONTH(SALES_DATE), REGION ) ) )
ORDER BY WEEK, DAY_WEEK, MONTH, REGION

```

da como resultado:

R1	R2	WEEK	DAY_WEEK	MONTH	REGION	UNITS_SOLD
GROUP 1	-	13	6	-	-	27
GROUP 1	-	13	7	-	-	46
GROUP 1	-	13	-	-	-	73
GROUP 1	-	14	1	-	-	31
GROUP 1	-	14	2	-	-	43
GROUP 1	-	14	-	-	-	74
GROUP 1	-	53	1	-	-	8
GROUP 1	-	53	-	-	-	8
-	GROUP 2	-	-	-	3 Manitoba	22
-	GROUP 2	-	-	-	3 Ontario-North	8
-	GROUP 2	-	-	-	3 Ontario-South	34
-	GROUP 2	-	-	-	3 Quebec	40
-	GROUP 2	-	-	-	3 -	104
-	GROUP 2	-	-	-	4 Manitoba	17
-	GROUP 2	-	-	-	4 Ontario-North	1
-	GROUP 2	-	-	-	4 Ontario-South	14
-	GROUP 2	-	-	-	4 Quebec	11
-	GROUP 2	-	-	-	4 -	43
-	GROUP 2	-	-	-	12 Manitoba	2
-	GROUP 2	-	-	-	12 Ontario-South	4
-	GROUP 2	-	-	-	12 Quebec	2
-	GROUP 2	-	-	-	12 -	8
-	GROUP 2	-	-	-	-	155
GROUP 1	-	-	-	-	-	155

Paso 4: Tenga en cuenta que puesto que R1 y R2 se utilizan en conjuntos de agrupación diferentes, siempre que R1 no sea nulo en el resultado, R2 es nulo y siempre que R2 sea no nulo en el resultado, R1 es nulo. Esto significa que

Ejemplos de conjuntos de agrupación, Cube y Rollup

puede consolidar estas columnas en una sola utilizando la función COALESCE. También puede utilizar esta columna en la cláusula ORDER BY para conservar el resultado de los dos conjuntos de agrupación juntos.

```

SELECT COALESCE(R1,R2) AS GROUP,
       WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       MONTH(SALES_DATE) AS MONTH,
       REGION, SUM(SALES) AS UNITS_SOLD
FROM SALES, (VALUES('GROUP 1','GROUP 2')) AS X(R1,R2)
GROUP BY GROUPING SETS ((R1, ROLLUP(WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE))),
(R2, ROLLUP( MONTH(SALES_DATE), REGION ) ) )
ORDER BY GROUP, WEEK, DAY_WEEK, MONTH, REGION;

```

da como resultado:

GROUP	WEEK	DAY_WEEK	MONTH	REGION	UNITS_SOLD
GROUP 1	13	6	-	-	27
GROUP 1	13	7	-	-	46
GROUP 1	13	-	-	-	73
GROUP 1	14	1	-	-	31
GROUP 1	14	2	-	-	43
GROUP 1	14	-	-	-	74
GROUP 1	53	1	-	-	8
GROUP 1	53	-	-	-	8
GROUP 1	-	-	-	-	155
GROUP 2	-	-	3	Manitoba	22
GROUP 2	-	-	3	Ontario-North	8
GROUP 2	-	-	3	Ontario-South	34
GROUP 2	-	-	3	Quebec	40
GROUP 2	-	-	3	-	104
GROUP 2	-	-	4	Manitoba	17
GROUP 2	-	-	4	Ontario-North	1
GROUP 2	-	-	4	Ontario-South	14
GROUP 2	-	-	4	Quebec	11
GROUP 2	-	-	4	-	43
GROUP 2	-	-	12	Manitoba	2
GROUP 2	-	-	12	Ontario-South	4
GROUP 2	-	-	12	Quebec	2
GROUP 2	-	-	12	-	8
GROUP 2	-	-	-	-	155

Ejemplo C8: El ejemplo siguiente ilustra la utilización de varias funciones de columna cuando se realiza un CUBE. El ejemplo también utiliza funciones de conversión del tipo de datos y el redondeo para producir resultados decimales con una precisión y escala razonables.

```

SELECT MONTH(SALES_DATE) AS MONTH,
       REGION,
       SUM(SALES) AS UNITS_SOLD,
       MAX(SALES) AS BEST_SALE,

```

Ejemplos de conjuntos de agrupación, Cube y Rollup

```

CAST(ROUND(AVG(DECIMAL(SALES)),2) AS DECIMAL(5,2)) AS AVG_UNITS_SOLD
FROM SALES
GROUP BY CUBE(MONTH(SALES_DATE),REGION)
ORDER BY MONTH, REGION

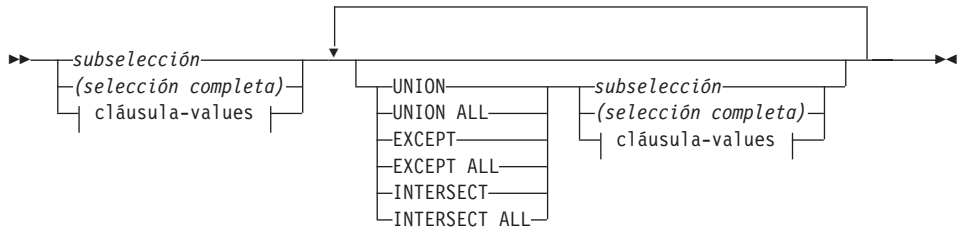
```

Da como resultado:

MONTH	REGION	UNITS_SOLD	BEST_SALE	AVG_UNITS_SOLD
3	Manitoba	22	7	3.14
3	Ontario-North	8	3	2.67
3	Ontario-South	34	14	4.25
3	Quebec	40	18	5.00
3	-	104	18	4.00
4	Manitoba	17	9	5.67
4	Ontario-North	1	1	1.00
4	Ontario-South	14	8	4.67
4	Quebec	11	8	5.50
4	-	43	9	4.78
12	Manitoba	2	2	2.00
12	Ontario-South	4	3	2.00
12	Quebec	2	1	1.00
12	-	8	3	1.60
-	Manitoba	41	9	3.73
-	Ontario-North	9	3	2.25
-	Ontario-South	52	14	4.00
-	Quebec	53	18	4.42
-	-	155	18	3.87

selección completa

selección completa



cláusula-values:



fila-valores:



La *selección completa* es un componente de la sentencia-select, la sentencia INSERT y la sentencia CREATE VIEW. También es un componente de algunos predicados que, a su vez, son componentes de una sentencia. Una *selección completa* que es un componente de un predicado se llama una *subconsulta*. Una *selección completa* que está encerrada entre paréntesis se llama a veces una subconsulta.

Los operadores de conjuntos UNION, EXCEPT e INTERSECT se corresponden a los operadores relacionales de unión, diferencia e intersección.

Una *selección completa* especifica una tabla resultante. Si no se utiliza un operador de conjunto, el resultado de la *selección completa* es el resultado de la subselección especificada o la cláusula-values.

cláusula-values

Obtiene una tabla resultante especificando los valores reales, utilizando expresiones, para cada columna de una fila de la tabla resultante. Pueden especificarse múltiples filas.

NULL sólo se puede utilizar con múltiples *filas-valores* y como mínimo una fila de la misma columna no debe ser NULL (SQLSTATE 42826).

Una *fila-valores* se especifica por:

- Una sola expresión para una tabla resultante de una sola columna o
- n expresiones (o NULL) separadas por comas y encerradas entre paréntesis, donde n es el número de columnas de la tabla resultante.

La cláusula VALUES de múltiples filas deben tener el mismo número de expresiones en cada *filas-valores* (SQLSTATE 42826).

A continuación encontrará ejemplos de cláusulas-valores y su significado.

VALUES (1),(2),(3)	- 3 filas de 1 columna
VALUES 1, 2, 3	- 3 filas de 1 columna
VALUES (1, 2, 3)	- 1 fila de 3 columnas
VALUES (1,21),(2,22),(3,23)	- 3 filas de 2 columnas

Una cláusula-valores que está compuesta de n *filas-valores*, RE₁ a RE_n, donde n es mayor que 1, es equivalente a

RE₁ UNION ALL RE₂ ... UNION ALL RE_n

Esto significa que las expresiones correspondientes de cada *fila-valores* deben poderse comparar (SQLSTATE 42825) y el tipo de datos resultante se basa en el apartado “Reglas para los tipos de datos del resultado” en la página 120.

UNION o UNION ALL

Obtiene una tabla resultante combinando las otras dos tablas resultantes (R1 y R2). Si se ha especificado UNION ALL, el resultado consta de todas las filas de R1 y de R2. Si se especifica UNION sin la opción ALL, el resultado consta de todas las filas de R1 o R2, con las filas duplicadas eliminadas. Sin embargo, en cualquier caso, todas las filas de la tabla UNION es una fila de R1 o una fila de R2.

EXCEPT o EXCEPT ALL

Obtiene una tabla resultante combinando las otras dos tablas resultantes (R1 y R2). Si se especifica EXCEPT ALL, el resultado consta de todas las filas que no tienen una fila correspondiente en R2, donde las filas duplicadas son significativas. Si se especifica EXCEPT sin la opción ALL, el resultado consta de todas las filas que están sólo en R1, y las filas duplicadas se eliminan del resultado de esta operación.

INTERSECT o INTERSECT ALL

Obtiene una tabla resultante combinando las otras dos tablas resultantes (R1 y R2). Si se especifica INTERSECT ALL, el resultado consta de todas las filas que están en R1 y en R2. Si se especifica INTERSECT sin la opción ALL, el resultado consta de todas las filas que están en R1 y en R2, con las filas duplicadas eliminadas.

selección completa

El número de columnas de las tablas resultantes R1 y R2 han de ser iguales (SQLSTATE 42826). Si no se especifica la palabra clave ALL, R1 y R2 no deben contener ninguna columna declarada de tipo serie de más de 255 bytes o cuyo tipo de datos sea LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB, DATALINK, un tipo diferenciado de estos tipos o un tipo estructurado (SQLSTATE 42907).

Las columnas del resultado tienen estos nombres:

- Si la columna *n* de R1 y la columna *n* de R2 tienen el mismo nombre de columna del resultado, la columna *n* de R tiene el nombre de la columna del resultado.
- Si la columna *n* de R1 y la columna *n* de R2 tienen nombres de columna del resultado diferente, se genera un nombre. Este nombre no se puede utilizar como nombre de columna en una cláusula ORDER BY o UPDATE.

El nombre generado se puede determinar ejecutando DESCRIBE de la sentencia de SQL y consultando el campo SQLNAME.

Dos filas son duplicadas entre sí si cada valor de la primera es igual al valor correspondiente de la segunda. (Para la determinación de duplicados, dos valores nulos se consideran iguales.)

Cuando se combinan múltiples operaciones en una expresión, las operaciones entre paréntesis se llevan a cabo primero. Si no hay paréntesis, las operaciones se llevan a cabo de izquierda a derecha a excepción de que todas las operaciones INTERSECT se efectúan antes que las operaciones UNION o EXCEPT.

En el ejemplo siguiente, los valores de las tablas R1 y R2 se muestran en la izquierda. Las otras cabeceras listadas muestran los valores como resultado de varias operaciones de conjunto en R1 y en R2.

R1	R2	UNION ALL	UNION	EXCEPT ALL	EXCEPT	INTER- SECT ALL	INTER- SECT
1	1	1	1	1	2	1	1
1	1	1	2	2	5	1	3
1	3	1	3	2		3	4
2	3	1	4	2		4	
2	3	1	5	4			
2	3	2		5			
3	4	2					
4		2					

R1	R2	UNION ALL	UNION	EXCEPT ALL	EXCEPT	INTER- SECT ALL	INTER- SECT
4		3					
5		3					
		3					
		3					
		3					
		4					
		4					
		4					
		5					

Para ver las reglas de cómo se determinan los tipos de datos de las columnas del resultado, consulte el apartado “Reglas para los tipos de datos del resultado” en la página 120.

Para ver las reglas sobre cómo se manejan las conversiones de las columnas de serie, consulte el apartado “Reglas para las conversiones de series” en la página 124.

Ejemplos de una selección completa

Ejemplo 1: Seleccione todas las columnas y filas de la tabla EMPLOYEE.

```
SELECT * FROM EMPLOYEE
```

Ejemplo 2: Liste los números de empleado (EMPNO) de todos los empleados de la tabla EMPLOYEE cuyo número de departamento (WORKDEPT) empiece por 'E' o que estén asignados a proyectos de la tabla EMP_ACT cuyo número de proyecto (PROJNO) sea igual a 'MA2100', 'MA2110' o 'MA2112'.

```
SELECT EMPNO
FROM EMPLOYEE
WHERE WORKDEPT LIKE 'E%'
UNION
SELECT EMPNO
FROM EMP_ACT
WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
```

Ejemplo 3: Haga la misma consulta que en el ejemplo 2 y, además, “identifique” las filas de la tabla EMPLOYEE con 'emp' y las filas de la tabla EMP_ACT con 'emp_act'. A diferencia del resultado del ejemplo 2, esta consulta puede devolver el mismo EMPNO más de una vez, identificando la tabla del que proviene mediante el “identificador” asociado.

Ejemplos de una selección completa

```
SELECT EMPNO, 'emp'  
FROM EMPLOYEE  
WHERE WORKDEPT LIKE 'E%'  
UNION  
SELECT EMPNO, 'emp_act' FROM EMP_ACT  
WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
```

Ejemplo 4: Realice la misma consulta que en el ejemplo 2, sólo que utilice UNION ALL para que no se elimine ninguna fila duplicada.

```
SELECT EMPNO  
FROM EMPLOYEE  
WHERE WORKDEPT LIKE 'E%'  
UNION ALL  
SELECT EMPNO  
FROM EMP_ACT  
WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
```

Ejemplo 5: Realice la misma consulta que en el ejemplo 3, sólo que esta vez incluya dos empleados adicionales que no están actualmente en ninguna tabla e identifíquelos como "new".

```
SELECT EMPNO, 'emp'  
FROM EMPLOYEE  
WHERE WORKDEPT LIKE 'E%'  
UNION  
SELECT EMPNO, 'emp_act'  
FROM EMP_ACT  
WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')  
UNION  
VALUES ('NEWAAA', 'new'), ('NEWBBB', 'new')
```

Ejemplo 6: Este ejemplo de EXCEPT genera todas las filas que están en T1 pero no en T2.

```
(SELECT * FROM T1)  
EXCEPT ALL  
(SELECT * FROM T2)
```

Si no hay ningún valor NULL implicado, este ejemplo devuelve los mismos resultados que

```
SELECT ALL *  
FROM T1  
WHERE NOT EXISTS (SELECT * FROM T2  
WHERE T1.C1 = T2.C1 AND T1.C2 = T2.C2 AND...)
```

Ejemplo 7: Este ejemplo de INTERSECT genera todas las filas que están en ambas tablas, T1 y T2, eliminando los duplicados.

```
(SELECT * FROM T1)  
INTERSECT  
(SELECT * FROM T2)
```

Ejemplos de una selección completa

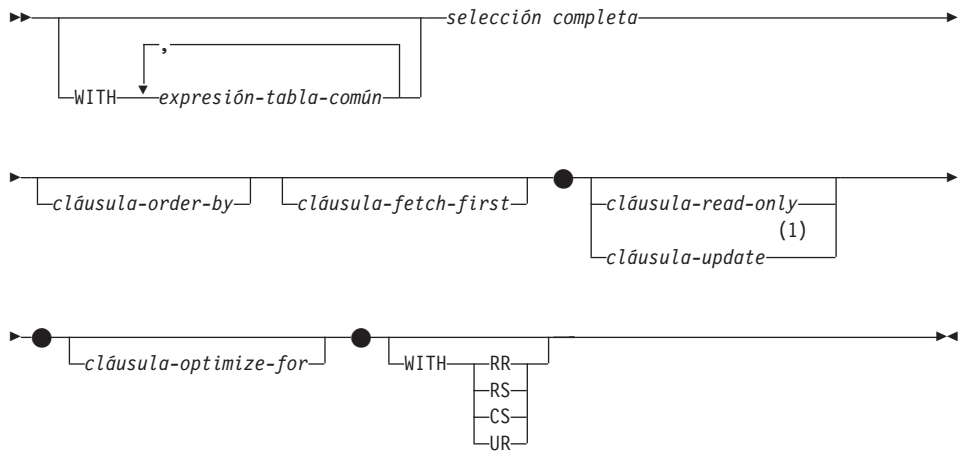
Si no hay valores NULL implicados, este ejemplo devuelve el mismo resultado que

```
SELECT DISTINCT * FROM T1
  WHERE EXISTS (SELECT * FROM T2
                WHERE T1.C1 = T2.C1 AND T1.C2 = T2.C2 AND...)
```

donde C1, C2, etcétera representan las columnas de T1 y T2.

sentencia-select

sentencia-select



Notas:

- 1 La cláusula-update y la cláusula-order-by no pueden especificarse en la misma sentencia-select.

La *sentencia-select* es la forma de una consulta que puede especificarse directamente en una sentencia DECLARE CURSOR o prepararse y después hacerse referencia en una sentencia DECLARE CURSOR. También puede emitirse mediante la utilización de sentencias de SQL dinámicas utilizando el procesador de la línea de mandatos (o herramientas similares) haciendo que se visualice una tabla resultante en la pantalla del usuario. En cualquier caso, la tabla especificada por una *sentencia-select* es el resultado de la selección completa.

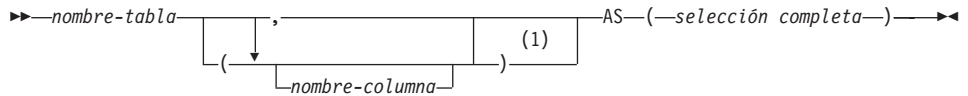
Una *sentencia-select* que haga referencia a un apodo no se puede especificar directamente en la sentencia DECLARE CURSOR.

La cláusula opcional WITH especifica el nivel de aislamiento en el que se ejecuta la sentencia select.

- RR - Lectura repetible
- RS - Estabilidad de lectura
- CS - Estabilidad del cursor
- UR - Lectura no confirmada

El nivel de aislamiento por omisión de la sentencia es el nivel de aislamiento del paquete en el que está enlazada la sentencia.

expresión-común-tabla



Notas:

- 1 Si una expresión de tabla común es recursiva o si la selección completa da como resultado nombres de columna duplicados, deben especificarse los nombres de columna.

Una *expresión de tabla común* permite la definición de una tabla resultante con un *nombre-tabla* que puede especificarse como un nombre de tabla en cualquier cláusula FROM de la selección completa que sigue. Se pueden especificar múltiples expresiones de tabla comunes a continuación de una sola palabra clave WITH. Cada expresión de tabla común especificada también puede referirse por nombre en la cláusula FROM de expresiones de tabla comunes subsiguientes.

Si se especifica una lista de columnas, debe constar de tantos nombres como el número de columnas de la tabla resultante de la selección completa. Cada *nombre-columna* debe ser exclusivo y no calificado. Si no se especifican estos nombres de columna, los nombres se obtienen de la lista de selección de la selección completa utilizada para definir la expresión de tabla común.

El *nombre-tabla* de una *expresión de tabla común* debe ser diferente de cualquier otro *nombre-tabla* de una expresión de tabla común de la misma sentencia (SQLSTATE 42726). Si se especifica la expresión de tabla común en una sentencia INSERT, el *nombre-tabla* no puede ser el mismo que el nombre de tabla o vista que es el objeto de la inserción (SQLSTATE 42726). Un *nombre-tabla* de una expresión de tabla común puede especificarse como nombre de tabla en cualquier cláusula FROM de toda la selección completa. Un *nombre-tabla* de una expresión de tabla común prevalece sobre cualquier tabla, vista oseudónimo existentes (en el catálogo) que tenga el mismo nombre calificado.

Si se define más de una expresión de tabla común en la misma sentencia, no están permitidas las referencias cíclicas entre las expresiones de tabla comunes (SQLSTATE 42835). Una *referencia cíclica* se produce cuando dos expresiones de tabla comunes *dt1* y *dt2* están creadas de tal manera que *dt1* hace referencia a *dt2* y *dt2* hace referencia a *dt1*.

La *expresión de tabla común* también es opcional antes de la selección completa en las sentencias CREATE VIEW e INSERT.

expresión-tabla-común

Se puede utilizar una *expresión de tabla común*:

- En lugar de una vista para evitar crear la vista (cuando no sea necesaria la utilización general de la vista y no se utilicen actualizaciones ni supresiones colocadas)
- Para permitir la agrupación por una columna que se obtiene de una subselección o función escalar que no es determinista o tiene una acción externa
- Cuando la tabla resultante deseada se basa en variables del lenguaje principal
- Cuando la misma tabla resultante necesite compartirse en una *selección completa*
- Cuando el resultado necesita obtenerse mediante recurrencia.

Si una *selección completa* de una expresión de tabla común contiene una referencia a sí misma en una cláusula FROM, la expresión de tabla común es *recursiva*. Las consultas que utilizan la recurrencia son útiles en las aplicaciones que permiten su uso, tales como la lista de material (BOM), sistemas de reservas y planificación de la red. Para ver un ejemplo, consulte el “Apéndice M. Ejemplo de recurrencia: Lista de material” en la página 1499.

Deben cumplirse las condiciones siguientes en una expresión de tabla común recursiva:

- Cada selección completa que forma parte del ciclo de repetición debe empezar por SELECT o SELECT ALL. La utilización de SELECT DISTINCT no está permitida (SQLSTATE 42925). Además, las uniones deben utilizar UNION ALL (SQLSTATE 42925).
- Los nombres de columna deben especificarse a continuación del *nombre-tabla* de la expresión de tabla común (SQLSTATE 42908).
- La primera selección completa de la primera unión (la selección completa de inicialización) no debe incluir ninguna referencia a ninguna columna de la expresión de tabla común de cualquier cláusula FROM (SQLSTATE 42836).
- Si se hace referencia a un nombre de columna de la expresión de tabla común en la selección completa repetida, el tipo de datos, longitud y página de códigos para la columna se determinan basándose en la selección completa de inicialización. La columna correspondiente de la selección completa recursiva debe tener el mismo tipo de datos y longitud que el tipo de datos y longitud determinados en base a la selección completa de inicialización y la página de códigos debe coincidir (SQLSTATE 42825). Sin embargo, para los tipos de serie de caracteres, la longitud de los dos tipos de datos puede diferir. En este caso, la columna de la selección completa recursiva debe tener una longitud que podría asignarse siempre a la longitud determinada de la selección completa de inicialización.

- Cada selección completa que forma parte del ciclo de repetición no debe incluir ninguna función de columna, cláusula-group-by ni cláusula-having (SQLSTATE 42836).
Las cláusulas FROM de estas selecciones completas pueden incluir como máximo una referencia a una expresión de tabla común que forme parte de un ciclo de repetición (SQLSTATE 42836).
- Las subconsultas (escalares o cuantificadas) no deben formar parte de ciclos de repetición (SQLSTATE 42836).

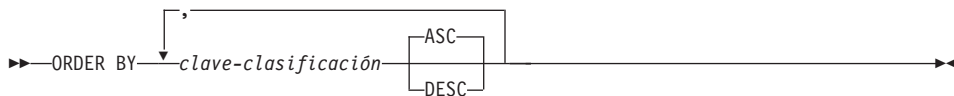
Cuando desarrolle expresiones de tabla comunes recursivas, recuerde que se puede crear un ciclo de repetición infinito (bucle). Compruebe que los ciclos de repetición terminen. Es muy importante si los datos implicados son cíclicos. Se espera que una expresión de tabla común recursiva incluya un predicado que impida un bucle infinito. Se espera que la expresión de tabla común recursiva incluya:

- Una selección completa recursiva, una columna de enteros incrementada por una constante.
- Un predicado en la cláusula where de la selección completa recursiva con el formato `col_contador < constante` o `col_contador < :var_lengprinc`.

Se emite un aviso si no se encuentra esta sintaxis en la expresión de tabla común recursiva (SQLSTATE 01605).

cláusula-order-by

cláusula-order-by



clave-clasificación:



La cláusula ORDER BY especifica una ordenación de las filas de la tabla resultante. Si se especifica una clasificación individual (una *clave-clasificación* con una dirección asociada), las filas se ordenan por los valores de dicha especificación de clasificación. Si se indica más de una especificación de clasificación, las filas se ordenan por los valores de la primera especificación de clasificación identificada, después por los valores de la segunda especificación de clasificación identificada, y así sucesivamente. La longitud de cada *clave-clasificación* no debe ser mayor que 255 caracteres, para una columna de caracteres, o 127 caracteres para una columna gráfica (SQLSTATE 42907), y su tipo de datos no puede ser LONG VARCHAR, LONG VARCHARIC, BLOB, CLOB, DBCLOB, DATALINK, un tipo diferenciado de cualquiera de estos tipos ni un tipo estructurado (SQLSTATE 42907).

Una columna con nombre de la lista de selección se puede identificar mediante una *clave-clasificación* que sea un *entero-simple* o un *nombre-columna-simple*. Una columna sin nombre de la lista de selección debe identificarse por un *entero-simple* o, en algunos casos, por una *expresión-clave-clasificación* que coincida con la expresión de la lista de selección (consulte los detalles de *expresión-clave-clasificación*). Una columna no tiene nombre si no se especifica la cláusula AS y se obtiene a partir de una constante, una expresión con operadores o una función.⁵⁴

La ordenación se realiza de acuerdo con las reglas de comparación descritas en el Capítulo 3. El valor nulo es superior a cualquier otro valor. Si la cláusula ORDER BY no ordena por completo las filas, se visualizan las filas con valores duplicados de todas las columnas identificadas en un orden arbitrario.

54. Las reglas para determinar el nombre de las columnas resultantes en una selección completa donde intervengan operadores de conjuntos (UNION, INTERSECT o EXCEPT) se pueden encontrar en “selección completa” en la página 512.

nombre-columna-simple

Normalmente identifica una columna de la tabla resultante. En este caso, *nombre-columna-simple* debe ser el nombre de columna de una columna con nombre de la lista de selección.

El *nombre-columna-simple* también puede identificar un nombre de columna de una tabla, vista o tabla anidada identificada en la cláusula FROM si la consulta es una subselección. Se produce un error si la subselección:

- especifica DISTINCT en la cláusula de selección (SQLSTATE 42822)
- produce un resultado agrupado y el *nombre-columna-simple* no es una *expresión-agrupación* (SQLSTATE 42803).

La determinación de qué columna se utiliza para ordenar el resultado se describe en el apartado "Nombres de columna en claves de clasificación" (consulte las "Notas" en la página 524).

entero-simple

Debe ser mayor que 0 y no ser superior al número de columnas de la tabla resultante (SQLSTATE 42805). El entero *n* identifica la columna *n* de la tabla resultante.

expresión-clave-clasificación

Una expresión que no es simplemente un nombre de columna ni una constante de enteros sin signo. La consulta a la que se aplica la ordenación debe ser una *subselección* para utilizar esta forma de clave-clasificación. La *expresión-clave-clasificación* no puede incluir una selección completa-escalar correlacionada (SQLSTATE 42703) ni una función con una acción externa (SQLSTATE 42845).

Cualquier nombre-columna de una *expresión-clave-clasificación* debe ajustarse a las reglas descritas bajo "Nombres de columna en claves de clasificación" (consulte las "Notas" en la página 524).

Existen unos cuantos casos especiales que restringen más las expresiones que se pueden especificar.

- DISTINCT se especifica en la cláusula SELECT de la subselección (SQLSTATE 42822).

La expresión-clave-clasificación debe coincidir exactamente con una expresión de la lista de selección de la subselección (las selecciones completas-escalares nunca se emparejan).

- La subselección está agrupada (SQLSTATE 42803).

La expresión-clave-clasificación puede:

- ser una expresión en la lista de selección de la subselección,
- incluir una *expresión-agrupación* de la cláusula GROUP BY de la subselección

cláusula-order-by

- incluir una función de columna, constante o variable del lenguaje principal.

ASC

Utiliza los valores de la columna en orden ascendente. Éste es el valor por omisión.

DESC

Utiliza los valores de la columna en orden descendente.

Notas

- **Nombres de columna en claves de clasificación:**

- El nombre de columna está calificado.

La consulta debe ser una *subselección* (SQLSTATE 42877). El nombre de columna debe identificar sin ambigüedades una columna de alguna tabla, vista o tabla anidada en la cláusula FROM de la subselección (SQLSTATE 42702). El valor de la columna se utiliza para calcular el valor de la especificación de clasificación.

- El nombre de columna no está calificado.

- La consulta es una subselección.

Si el nombre de columna es idéntico al nombre de más de una columna de la tabla resultante, el nombre de columna debe identificar sin ambigüedades una columna de alguna tabla, vista o tabla anidada en la cláusula FROM de la subselección de orden (SQLSTATE 42702).

Si el nombre de columna es idéntico a una columna, dicha columna se utiliza para calcular el valor de la especificación de clasificación. Si el nombre de columna no es idéntico a una columna de la tabla resultante, debe identificar sin ambigüedades una columna de alguna tabla, vista o tabla anidada en la cláusula FROM de la selección completa de la sentencia-select (SQLSTATE 42702).

- La consulta no es una subselección (incluye operaciones de conjuntos como la unión, excepción o intersección).

El nombre de columna no debe ser idéntico al nombre de más de una columna de la tabla resultante (SQLSTATE 42702). El nombre de columna debe ser idéntico a exactamente una columna de la tabla resultante (SQLSTATE 42707) y esta columna se utiliza para calcular el valor de la especificación de clasificación.

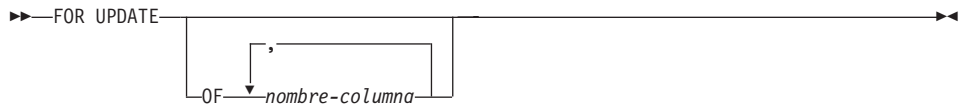
Consulte el apartado “Calificadores de nombres de columna para evitar ambigüedad” en la página 147 para obtener más información sobre nombres de columna no calificados.

- **Límites:** La utilización de una *expresión-clave-clasificación* o un *nombre-columna-simple* donde la columna no está en la lista de selección puede dar como resultado la adición de la columna o expresión a la tabla temporal utilizada para clasificación. Esto puede dar como resultado que se

alcance el límite del número de columnas de una tabla o el límite en el tamaño de una fila de una tabla. Si se exceden estos límites se producirá un error si es necesaria una tabla temporal para realizar la operación de clasificación.

cláusula-update

cláusula-update



La cláusula FOR UPDATE identifica las columnas que se pueden actualizar en una sentencia UPDATE con posición posterior. Cada *nombre-columna* debe estar sin calificar y debe identificar una columna de la tabla o vista identificada en la primera cláusula FROM de la selección completa. Si la cláusula FOR UPDATE se especifica sin nombres de columna, se incluyen todas las columnas actualizables de la tabla o vista identificadas en la primera cláusula FROM de la selección completa.

La cláusula FOR UPDATE no puede utilizarse si es verdadera una de las siguientes situaciones:

- El cursor asociado con la sentencia-select no se puede suprimir (consulte las “Notas” en la página 979).
- Una de las columnas seleccionadas no es una columna actualizable de una tabla del catálogo y la cláusula FOR UPDATE no se ha utilizado para excluir dicha columna.

cláusula-read-only



La cláusula FOR READ ONLY indica que la tabla resultante es de sólo lectura y que, por lo tanto, no se puede hacer referencia al cursor en las sentencias UPDATE con posición y DELETE. FOR FETCH ONLY tiene el mismo significado.

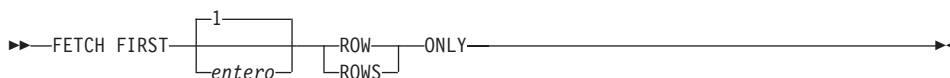
Algunas tablas resultantes son de sólo lectura por naturaleza. (Por ejemplo, una tabla basada en una vista de sólo lectura.) Se puede seguir especificando FOR READ ONLY para dichas tablas, pero la especificación no surtirá efecto.

Para las tablas resultantes en las que están permitidas las actualizaciones y supresiones, la especificación de FOR READ ONLY (o FOR FETCH ONLY) posiblemente mejorará el rendimiento de las operaciones FETCH permitiendo al gestor de bases de datos realizar el bloqueo y evitar bloqueos exclusivos. Por ejemplo, en los programas que contienen sentencias del SQL dinámico sin la cláusula FOR READ ONLY u ORDER BY, el gestor de bases de datos puede abrir cursores como si se hubiese especificado la cláusula FOR UPDATE. Por lo tanto, se recomienda utilizar la cláusula FOR READ ONLY para mejorar el rendimiento excepto en los casos en que se utilizarán las consultas en sentencias UPDATE con posición o DELETE.

No se debe hacer referencia a una tabla resultante de sólo lectura en una sentencia UPDATE con posición o DELETE, ya sea de sólo lectura por naturaleza o especificada como FOR READ ONLY (FOR FETCH ONLY). Consulte el apartado “DECLARE CURSOR” en la página 976 para obtener más información acerca de los cursores de sólo lectura y actualizables.

cláusula-fetch-first

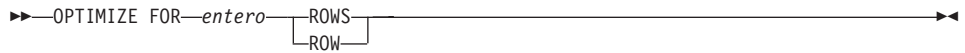
cláusula-fetch-first



La *cláusula-fetch-first* establece el número máximo de filas que pueden recuperarse. Indica al gestor de bases de datos que la aplicación no recuperará más de *entero* filas, cualquiera que sea el número de filas que pueda haber en la tabla resultante cuando no se especifica esta cláusula. Cualquier intento de recuperar más filas que el número indicado por *entero* se trata de la misma manera que un fin de datos normal (SQLSTATE 02000). El valor de *entero* debe ser un entero positivo, distinto de cero.

Limitar la tabla resultante a las primeras *entero* filas puede mejorar el rendimiento. El gestor de bases de datos detendrá el proceso de la consulta una vez que haya determinado las *entero* primeras filas. Si se especifican la *cláusula-fetch-first* y la *cláusula-optimize-for*, se utilizará el valor *entero* más bajo de estas cláusulas para influir en el tamaño del almacenamiento intermedio de comunicaciones. Los valores se tienen en cuenta de forma independiente por motivos de optimización.

cláusula-optimize-for



La cláusula `OPTIMIZE FOR` pide el proceso especial de la *sentencia de selección*. Si se omite la cláusula, se supone que se recuperarán todas las filas de la tabla resultante; si se especifica, se supone que el número de filas recuperado probablemente no excederán de n donde n es el valor para *entero*. El valor de n debe ser un entero positivo. La utilización de la cláusula `OPTIMIZE FOR` influye en la optimización de la consulta basándose en la suposición de que se recuperarán n filas. Además, cuando los cursores están bloqueados, esta cláusula afecta al número de filas que se devuelven en cada bloque (es decir, no se devolverán más de n filas en cada bloque). Si se especifican la *cláusula-fetch-first* y la *cláusula-optimize-for*, se utilizará el valor entero menor de estas cláusulas para determinar el tamaño del almacenamiento intermedio de comunicaciones. Los valores se consideran de forma independiente con fines de optimización.

Esta cláusula no limita el número de filas que se pueden recuperar ni afecta al resultado de ninguna otra manera que no sea en el rendimiento. La utilización de `OPTIMIZE FOR n ROWS` puede mejorar el rendimiento si no se recuperan más de n filas, pero puede reducir el rendimiento si se recuperan más de n filas.

Si el valor de n multiplicado por el tamaño de la fila, excede el tamaño del almacenamiento intermedio de comunicaciones⁵⁵ La cláusula `OPTIMIZE FOR` no tendrá ningún efecto sobre los almacenamientos intermedios de datos.

55. El tamaño del almacenamiento intermedio de comunicaciones está definido por el parámetro de configuración `RQRIOLBK` o `ASLHEAPSZ`. Vea el manual *Administration Guide* para obtener detalles.

Ejemplos de una sentencia-select

Ejemplos de una sentencia-select

Ejemplo 1: Seleccione todas las columnas y filas de la tabla EMPLOYEE.

```
SELECT * FROM EMPLOYEE
```

Ejemplo 2: Seleccione el nombre del proyecto (PROJNAME), fecha de inicio (PRSTDATE) y fecha de finalización (PRENDATE) de la tabla PROJECT. Ordene la tabla resultante por la fecha de finalización con las fechas más recientes primero.

```
SELECT PROJNAME, PRSTDATE, PRENDATE  
FROM PROJECT  
ORDER BY PRENDATE DESC
```

Ejemplo 3: Seleccione el número de departamento (WORKDEPT) y el salario promedio de departamento (SALARY) para todos los departamentos de la tabla EMPLOYEE. Ordene la tabla resultante por orden ascendente por el salario promedio de departamento.

```
SELECT WORKDEPT, AVG(SALARY)  
FROM EMPLOYEE  
GROUP BY WORKDEPT  
ORDER BY 2
```

Ejemplo 4: Declare un cursor llamado UP_CUR para utilizarlo en un programa C para actualizar las columnas de fecha de inicio (PRSTDATE) y de fecha de finalización (PRENDATE) en la tabla PROJECT. El programa debe recibir los dos valores junto con el valor de número del proyecto (PROJNO) para cada fila.

```
EXEC SQL DECLARE UP_CUR CURSOR FOR  
SELECT PROJNO, PRSTDATE, PRENDATE  
FROM PROJECT  
FOR UPDATE OF PRSTDATE, PRENDATE;
```

Ejemplo 5: Este ejemplo denomina a la expresión SAL+BONUS+COMM como TOTAL_PAY

```
SELECT SALARY+BONUS+COMM AS TOTAL_PAY  
FROM EMPLOYEE  
ORDER BY TOTAL_PAY
```

Ejemplo 6: Determine el número de empleado y el salario de los representantes de ventas junto con el salario promedio y el número total de empleados de sus departamentos. También, liste el salario promedio del departamento con el salario promedio más alto.

La utilización de una expresión de tabla común para este caso ahorra la actividad de crear una vista DINFO como una vista normal. Durante la preparación de la sentencia, se evita el acceso al catálogo para la vista y,

debido al contexto del resto de la selección completa, sólo se han de tener en cuenta las filas para el departamento de representantes de ventas para la vista.

```
WITH
DINFO (DEPTNO, AVGSALARY, EMPCOUNT) AS
    (SELECT OTHERS.WORKDEPT, AVG(OTHERS.SALARY), COUNT(*)
     FROM EMPLOYEE OTHERS
     GROUP BY OTHERS.WORKDEPT
    ),
DINFOMAX AS
    (SELECT MAX(AVGSALARY) AS AVGMAX FROM DINFO)
SELECT THIS_EMP.EMPNO, THIS_EMP.SALARY,
        DINFO.AVGSALARY, DINFO.EMPCOUNT, DINFOMAX.AVGMAX
FROM EMPLOYEE THIS_EMP, DINFO, DINFOMAX
WHERE THIS_EMP.JOB = 'SALESREP'
AND THIS_EMP.WORKDEPT = DINFO.DEPTNO
```

Ejemplos de una sentencia-select

Capítulo 6. Sentencias de SQL

Este capítulo contiene los diagramas de sintaxis, descripciones semánticas, reglas y ejemplos de la utilización de las sentencias de SQL.

Tabla 20. Sentencias de SQL

Sentencia de SQL	Función	Página
ALTER BUFFERPOOL	Cambia la definición de una agrupación de almacenamientos intermedios.	545
ALTER NICKNAME	Cambia la definición de un apodo.	548
ALTER NODEGROUP	Cambia la definición de un grupo de nodos.	552
ALTER SEQUENCE	Cambia la definición de una secuencia.	556
ALTER SERVER	Cambia la definición de un servidor.	561
ALTER TABLE	Cambia la definición de una tabla.	565
ALTER TABLESPACE	Cambia la definición de un espacio de tablas.	597
ALTER TYPE (Estructurado)	Cambia la definición de un tipo estructurado.	603
ALTER USER MAPPING	Cambia la definición de una correlación de autorizaciones de usuario.	610
ALTER VIEW	Cambia la definición de una vista modificando una columna de tipo de referencia para añadir un ámbito.	613
BEGIN DECLARE SECTION	Marca el principio de una sección de declaración de variables del lenguaje principal.	615
CALL	Invoca un procedimiento almacenado.	617
CLOSE	Cierra un cursor.	626
COMMENT	Sustituye o añade un comentario a la descripción de un objeto.	628
COMMIT	Finaliza una unidad de trabajo y confirma los cambios que esa unidad de trabajo ha realizado en la base de datos.	640
SQL compuesto (dinámico)	Combina una o más sentencias de SQL diferentes en un bloque dinámico.	642
SQL compuesto (incorporado)	Combina una o varias sentencias de SQL para formar un bloque ejecutable.	648
CONNECT (Tipo 1)	Conecta a un servidor de aplicaciones según las reglas para una unidad de trabajo remota.	653
CONNECT (Tipo 2)	Conecta a un servidor de aplicaciones según las reglas para la unidad de trabajo distribuida dirigida por aplicación.	662
CREATE ALIAS	Define un seudónimo para una tabla, vista u otro seudónimo.	670

Tabla 20. Sentencias de SQL (continuación)

Sentencia de SQL	Función	Página
CREATE BUFFERPOOL	Crea una nueva agrupación de almacenamientos intermedios.	673
CREATE DISTINCT TYPE	Define un tipo de datos diferenciado.	677
CREATE EVENT MONITOR	Especifica sucesos de la base de datos que se han de supervisar.	684
CREATE FUNCTION	Registra una función definida por el usuario.	695
CREATE FUNCTION (Escalar externa)	Registra una función escalar externa definida por el usuario.	696
CREATE FUNCTION (Tabla externa)	Registra una función de tabla externa definida por el usuario.	723
CREATE FUNCTION (Tabla externa OLE DB)	Registra una función de tabla externa OLE DB definida por el usuario.	741
CREATE FUNCTION (fuente o plantilla)	Registra una función derivada definida por el usuario.	750
CREATE FUNCTION (SQL, escalar, de tabla o de fila)	Registra y define una función SQL definida por el usuario.	761
CREATE FUNCTION MAPPING	Define una correlación de funciones.	769
CREATE INDEX	Define un índice para una tabla.	774
CREATE INDEX EXTENSION	Define un objeto de extensión para su uso con índices sobre tablas con columnas de tipo estructurado o diferenciado.	782
CREATE METHOD	Asocia un cuerpo de método con una especificación de método definida previamente.	790
CREATE NICKNAME	Define un apodo.	795
CREATE NODEGROUP	Define un grupo de nodos.	801
CREATE PROCEDURE	Registra un procedimiento almacenado.	804
CREATE SCHEMA	Define un esquema.	822
CREATE SEQUENCE	Define una secuencia.	826
CREATE SERVER	Define una fuente de datos para una base de datos federada.	831
CREATE TABLE	Define una tabla.	835
CREATE TABLESPACE	Define un espacio de tablas.	893
CREATE TRANSFORM	Define funciones de transformación.	903
CREATE TRIGGER	Define un desencadenante.	910
CREATE TYPE (Estructurado)	Define un tipo de datos estructurado.	923
CREATE TYPE MAPPING	Define una correlación entre tipos de datos.	950
CREATE USER MAPPING	Define una correlación entre autorizaciones de usuario.	955

Tabla 20. Sentencias de SQL (continuación)

Sentencia de SQL	Función	Página
CREATE VIEW	Define una vista de una o más tablas, vistas o apodos.	957
CREATE WRAPPER	Registra un reiniciador.	974
DECLARE CURSOR	Define un cursor SQL.	976
DECLARE GLOBAL TEMPORARY TABLE	Define la Tabla Temporal Global.	982
DELETE	Suprime una o más filas de una tabla.	991
DESCRIBE	Describe las columnas del resultado de una sentencia SELECT preparada.	997
DISCONNECT	Finaliza una o más conexiones cuando no hay ninguna unidad de trabajo activa.	1002
DROP	Suprime objetos de la base de datos.	1005
END DECLARE SECTION	Marca el final de una sección de declaración de variables del lenguaje principal.	1034
EXECUTE	Ejecuta una sentencia de SQL preparada.	1036
EXECUTE IMMEDIATE	Prepara y ejecuta una sentencia de SQL.	1042
EXPLAIN	Captura información acerca del plan de acceso elegido.	1045
FETCH	Asigna valores de una fila a variables del lenguaje principal.	1050
FLUSH EVENT MONITOR	Graba el almacenamiento intermedio interno activo de un supervisor de sucesos.	1054
FREE LOCATOR	Elimina la asociación entre una variable localizadora y su valor.	1055
GRANT (autorizaciones de base de datos)	Otorga autorizaciones sobre toda una base de datos.	1056
GRANT (privilegios de índice)	Otorga el privilegio CONTROL en índices en la base de datos.	1059
GRANT (privilegios de paquete)	Otorga privilegios para paquetes de la base de datos.	1061
GRANT (privilegios de esquema)	Otorga privilegios para un esquema.	1064
GRANT (privilegios de secuencia)	Otorga privilegios en una secuencia.	1067
GRANT (Privilegios de servidor)	Otorga privilegios para consultar una fuente de datos específica.	1068
GRANT (privilegios de apodo, vista o tabla)	Otorga privilegios para tablas, vistas y apodos.	1070
GRANT (privilegios para espacios de tablas)	Otorga privilegios para un espacio de tablas.	1078

Tabla 20. Sentencias de SQL (continuación)

Sentencia de SQL	Función	Página
INCLUDE	Inserta código o declaraciones en un programa fuente.	1080
INSERT	Inserta una o más filas en una tabla.	1082
LOCK TABLE	Impide que los procesos simultáneos cambien una tabla o impide que los procesos simultáneos utilicen una tabla.	1092
OPEN	Prepara un cursor que se utilizará para recuperar valores cuando se emita la sentencia FETCH.	1094
PREPARE	Prepara una sentencia de SQL (con parámetros opcionales) para su ejecución.	1099
REFRESH TABLE	Renueva los datos de una tabla de resumen.	1110
RELEASE (Conexión)	Coloca una o más conexiones en el estado pendiente de liberación.	1111
RELEASE SAVEPOINT	Libera un punto de salvar dentro de una transacción.	1113
RENAME TABLE	Cambia el nombre de una tabla existente.	1114
RENAME TABLESPACE	Cambia el nombre de un espacio de tablas existente.	1116
REVOKE (autorizaciones de bases de datos)	Revoca las autorizaciones de toda una base de datos.	1118
REVOKE (privilegios de índice)	Revoca el privilegio CONTROL en índices determinados.	1122
REVOKE (privilegios de paquete)	Revoca los privilegios de paquetes determinados en la base de datos.	1124
REVOKE (privilegios de esquema)	Revoca privilegios para un esquema.	1127
REVOKE (Privilegios de servidor)	Revoca privilegios para consultar una fuente de datos específica.	1129
REVOKE (privilegios de apodo, vista o tabla)	Revoca privilegios para determinadas tablas, vistas o apodos.	1131
REVOKE (privilegios para espacios de tablas)	Revoca el privilegio de utilización (USE) para un espacio de tablas determinado.	1137
ROLLBACK	Termina una unidad de trabajo y restituye los cambios realizados por dicha unidad de trabajo.	1139
SAVEPOINT	Define un punto de salvar dentro de una transacción.	1142
SELECT INTO	Especifica una tabla resultante de no más de una fila y asigna los valores a variables del lenguaje principal.	1145
SET CONNECTION	Cambia el estado de una conexión de inactivo a actual, haciendo que la ubicación especificada sea el servidor actual.	1147
SET CURRENT DEFAULT TRANSFORM GROUP	Cambia el valor del registro especial CURRENT DEFAULT TRANSFORM GROUP.	1149

Tabla 20. Sentencias de SQL (continuación)

Sentencia de SQL	Función	Página
SET CURRENT DEGREE	Cambia el valor del registro especial CURRENT DEGREE.	1151
SET CURRENT EXPLAIN MODE	Cambia el valor del registro especial CURRENT EXPLAIN MODE.	1153
SET CURRENT EXPLAIN SNAPSHOT	Cambia el valor del registro especial CURRENT EXPLAIN SNAPSHOT.	1155
SET CURRENT PACKAGESET	Establece el nombre de esquema para la selección de paquetes.	1157
SET CURRENT QUERY OPTIMIZATION	Cambia el valor del registro especial CURRENT QUERY OPTIMIZATION.	1159
SET CURRENT REFRESH AGE	Cambia el valor del registro especial CURRENT REFRESH AGE.	1162
SET ENCRYPTION PASSWORD	Establece la contraseña para el cifrado.	1164
SET EVENT MONITOR STATE	Activa o desactiva un supervisor de sucesos.	1166
SET INTEGRITY	Establece el estado pendiente de comprobación y comprueba los datos para las violaciones de restricciones.	1168
SET PASSTHRU	Abre una sesión para someter SQL nativo de fuente de datos directamente a la fuente de datos.	1179
SET PATH	Cambia el valor del registro especial CURRENT PATH.	1181
SET SCHEMA	Cambia el valor del registro especial CURRENT SCHEMA.	1184
SET SERVER OPTION	Establece valores de opciones del servidor.	1186
SET variable	Asigna valores a variables de transición NEW.	1188
UPDATE	Actualiza los valores de una o varias columnas en una o más filas de una tabla.	1193
VALUES INTO	Especifica una tabla resultante de no más de una fila y asigna los valores a variables del lenguaje principal.	1205
WHENEVER	Define las acciones que se han de tomar sobre la base de los códigos de retorno SQL.	1207

Cómo se invocan las sentencias de SQL

Las sentencias de SQL que se describen en este capítulo se clasifican en *ejecutables* y *no ejecutables*. La sección *Invocación* de la descripción de cada sentencia indica si la sentencia es ejecutable o no.

Una *sentencia ejecutable* puede invocarse de cuatro maneras:

- Incorporada en un programa de aplicación
- Incorporada en un procedimiento SQL.

- Preparada y ejecutada dinámicamente
- Emitida interactivamente

Nota: Las sentencias incorporadas en REXX se preparan y ejecutan dinámicamente.

Se pueden utilizar algunos o todos estos métodos, dependiendo de la sentencia. La sección *Invocación* de la descripción de cada sentencia indica qué métodos se pueden utilizar.

Una *sentencia no ejecutable* sólo puede estar incorporada en un programa de aplicación.

Además de las sentencias descritas en este capítulo, existe una construcción más de sentencia de SQL: la *sentencia-select*. (Vea “sentencia-select” en la página 518). No se incluye en este capítulo porque se utiliza de manera diferente a las demás sentencias.

Una *sentencia-select* se puede invocar de tres maneras:

- Incorporada en DECLARE CURSOR y ejecutada implícitamente por OPEN, FETCH y CLOSE
- Preparada dinámicamente, con referencia en DECLARE CURSOR y ejecutada implícitamente por OPEN, FETCH y CLOSE
- Emitida interactivamente.

Los dos primeros métodos se denominan, respectivamente, invocación *estática* e invocación *dinámica* de la *sentencia-select*.

Los distintos métodos de invocación de una sentencia de SQL se explican más abajo con más detalle. La explicación de cada método incluye el mecanismo de ejecución, interacción con variables del lenguaje principal y la comprobación de si la ejecución se realizó satisfactoriamente o no.

Incorporación de una sentencia en un programa de aplicación

Las sentencias de SQL se pueden incluir en un programa fuente que se someterá al precompilador. Se dice que dichas sentencias se *incorporan* al programa. Una sentencia incorporada se puede colocar en cualquier lugar del programa donde esté permitida una sentencia del lenguaje principal. Cada sentencia incorporada debe ir precedida por las palabras clave EXEC y SQL.

Sentencias ejecutables

Una sentencia ejecutable incorporada en un programa de aplicación se ejecuta cada vez que se ejecutaría una sentencia del lenguaje principal si se especificase en el mismo lugar. Por lo tanto, una sentencia dentro de un bucle

se ejecuta cada vez que se ejecuta el bucle, y una sentencia dentro de una construcción condicional sólo se ejecuta cuando se satisface la condición.

Una sentencia incorporada puede contener referencias a variables del lenguaje principal. Una variable del lenguaje principal a la que se hace referencia de esta manera puede utilizarse de dos maneras:

- Como entrada (el valor actual de la variable del lenguaje principal se utiliza en la ejecución de la sentencia)
- Como salida (se asigna un nuevo valor a la variable como resultado de la ejecución de la sentencia).

En particular, todas las referencias a variables del lenguaje principal en expresiones y predicados se sustituyen de manera efectiva por los valores actuales de las variables; es decir, las variables se utilizan como entrada. El tratamiento de otras referencias se describe individualmente para cada sentencia.

Todas las sentencias ejecutables deben ir seguidas de una prueba de un código de retorno SQL. Por otro lado, la sentencia WHENEVER (que es en sí no ejecutable) puede utilizarse para cambiar el flujo de control inmediatamente después de la ejecución de la sentencia incorporada.

Todos los objetos a los que las sentencias DML hacen referencia deben existir cuando se enlazan las sentencias con una DB2 Universal Database.

Sentencias no ejecutables

Una sentencia no ejecutable incorporada sólo se procesa por el precompilador. El precompilador informa de cualquier error encontrado en la sentencia. La sentencia no se procesa *nunca* durante la ejecución del programa. Por lo tanto, dichas sentencias no deben ir seguidas por una prueba de un código de retorno SQL.

Incorporación de una sentencia a un procedimiento SQL

Se pueden incorporar sentencias al cuerpo de un procedimiento SQL correspondiente a la sentencia CREATE PROCEDURE. Se dice que dichas sentencias se *incorporan* al procedimiento SQL. Las sentencias que se pueden incorporar a un procedimiento SQL se especifican en “Sentencia de procedimiento SQL” en la página 1210. A diferencia de las sentencias incorporadas a una aplicación, no es necesario que la sentencia de SQL esté precedida por una palabra clave. Siempre que una descripción de sentencia de SQL hace referencia a una *variable del lenguaje principal*, se puede utilizar una *variable de SQL* al incorporar la sentencia a un procedimiento SQL.

Preparación y ejecución dinámicas

Un programa de aplicación puede construir una sentencia de SQL en la forma de una serie de caracteres colocada en una variable de lenguaje principal. En

general, la sentencia se construye a partir de algunos datos disponibles para el programa (por ejemplo, la entrada de una estación de trabajo). La sentencia (distinta a una sentencia-select) construida de esta manera se puede preparar para la ejecución mediante la sentencia (incorporada) PREPARE y se ejecuta mediante la sentencia (incorporada) EXECUTE. Por otro lado, la sentencia (incorporada) EXECUTE IMMEDIATE puede utilizarse para preparar y ejecutar una sentencia en un paso.

Una sentencia que se va a preparar dinámicamente no debe contener referencias a variables del lenguaje principal. En su lugar puede contener marcadores de parámetros. (Consulte las reglas relativas a los marcadores de parámetros en el apartado “PREPARE” en la página 1099.) Cuando se ejecuta la sentencia preparada, los marcadores de parámetros se sustituyen de manera efectiva por los valores actuales de las variables del lenguaje principal especificadas en la sentencia EXECUTE. (Consulte las reglas relativas a esta sustitución en el apartado “EXECUTE” en la página 1036.) Una vez preparada, una sentencia se puede ejecutar varias veces con distintos valores de las variables del lenguaje principal. Los marcadores de parámetros no están permitidos en EXECUTE IMMEDIATE.

La ejecución satisfactoria o no satisfactoria de la sentencia viene indicada por el valor de un código de retorno SQL en la SQLCA después de la sentencia EXECUTE (o EXECUTE IMMEDIATE). El código de retorno SQL debe comprobarse tal como se describe arriba. Vea “Códigos de retorno de SQL” en la página 542 para obtener más información.

Invocación estática de una sentencia-select

Se puede incluir una *sentencia-select* como parte de la sentencia (no ejecutable) DECLARE CURSOR. Dicha sentencia se ejecuta cada vez que se abre el cursor mediante la sentencia (incorporada) OPEN. Después de abrir el cursor, se puede recuperar la tabla resultante, fila a fila, por ejecuciones sucesivas de la sentencia FETCH.

Cuando se utiliza de esta manera, la *sentencia-select* puede contener referencias a variables del lenguaje principal. Estas referencias se sustituyen de manera efectiva por los valores que las variables tienen en el momento de la ejecución de OPEN.

Invocación dinámica de una sentencia-select

Un programa de aplicación puede construir dinámicamente una *sentencia-select* en la forma de una serie de caracteres colocada en una variable del lenguaje principal. En general, la sentencia se construye a partir de algunos datos disponibles para el programa (por ejemplo, una consulta obtenida de una estación de trabajo). La sentencia construida de esta manera se puede preparar para su ejecución mediante la sentencia (incorporada) PREPARE y se puede hacer referencia por una sentencia (no ejecutable) DECLARE CURSOR.

Entonces la sentencia se ejecuta cada vez que se abre el cursor por medio de la sentencia (incorporada) OPEN. Después de abrir el cursor, se puede recuperar la tabla resultante, de fila en fila, mediante ejecuciones sucesivas de la sentencia FETCH.

Cuando se utiliza de esta manera, la *sentencia-select* no debe contener referencias a variables del lenguaje principal. Puede contener marcadores de parámetros en su lugar. (Consulte el apartado "PREPARE" en la página 1099 para conocer las reglas referentes a los marcadores de parámetros.) Los marcadores de parámetros se sustituyen de manera efectiva por los valores de las variables del lenguaje principal especificadas en la sentencia OPEN. (Consulte las reglas relativas a esta sustitución en el apartado "OPEN" en la página 1094.)

Invocación interactiva

Posibilidad de entrar sentencias de SQL desde una estación de trabajo como parte de la arquitectura del gestor de bases de datos. Se dice que una sentencia entrada así se emite interactivamente.

Una sentencia emitida interactivamente debe ser una sentencia ejecutable que no contenga marcadores de parámetros ni referencias a las variables del lenguaje principal, porque esto sólo tiene sentido en el contexto de un programa de aplicación.

Utilización de SQL con otros sistemas principales

La sintaxis de las sentencias de SQL presenta pequeñas variaciones entre los diversos tipos de sistemas principales (DB2 para z/OS, DB2 para iSeries, DB2 Universal Database). Independientemente de que las sentencias de SQL de una aplicación sean estáticas o dinámicas, asegúrese de que las sentencias de SQL y las opciones de precompilación/enlace se soporten en los sistemas de bases de datos a los que accederá la aplicación, si dicha aplicación va a acceder a diferentes sistemas principales de bases de datos.

Se puede encontrar información adicional sobre las sentencias de SQL utilizadas en otros sistemas principales en el manual DB2 Universal Database for iSeries SQL Reference y el manual DB2 Universal Database for OS/390 and z/OS SQL Reference.

Códigos de retorno de SQL

Un programa de aplicación que contiene sentencias de SQL ejecutables puede utilizar los valores SQLCODE o SQLSTATE para manejar los códigos de retorno de las sentencias de SQL. Hay dos maneras para que la aplicación acceda a estos valores.

- Incluir una estructura llamada SQLCA. Se proporciona una SQLCA automáticamente en REXX. En otros lenguajes, la SQLCA se puede obtener utilizando la sentencia INCLUDE SQLCA.

La SQLCA incluye una variable entera llamada SQLCODE y una variable de serie de caracteres llamada SQLSTATE.

- Cuando se especifica LANGLEVEL SQL92E como opción de precompilación, puede declararse una variable SQLCODE o SQLSTATE en la sección de declaración de SQL del programa. Si ninguna de estas variables está declarada en la sección de declaración de SQL, se supone que una variable SQLCODE está declarada en algún otro lugar del programa. Cuando se utiliza LANGLEVEL SQL92E, el programa no debe tener una sentencia INCLUDE SQLCA.

De manera ocasional, se mencionan condiciones de aviso además de las condiciones de error con respecto a los códigos de retorno. Un SQLCODE de aviso es un valor positivo y un SQLSTATE de aviso tiene los dos primeros caracteres establecidos en '01'.

SQLCODE

El gestor de bases de datos establece un SQLCODE después de ejecutar cada sentencia de SQL. Todos los gestores de bases de datos se ajustan al estándar SQL ISO/ANSI, tal como sigue:

- Si SQLCODE = 0 y SQLWARN0 está en blanco, la ejecución ha sido satisfactoria.
- Si SQLCODE = 100, no se ha encontrado “ningún dato”. Por ejemplo, una sentencia FETCH no ha devuelto ningún dato, porque el cursor estaba situado después de la última fila de la tabla resultante.
- Si SQLCODE > 0 y no = 100, la ejecución ha sido satisfactoria con un aviso.
- Si SQLCODE = 0 y SQLWARN0 = 'W', la ejecución ha sido satisfactoria; sin embargo, se han establecido uno o más indicadores de aviso. Consulte el “Apéndice B. Comunicaciones SQL (SQLCA)” en la página 1261 para obtener más detalles.
- Si SQLCODE < 0, la ejecución no ha sido satisfactoria.

El significado de los valores SQLCODE que no sean 0 ni 100 son específicos del producto. Consulte la publicación *Consulta de mensajes* para conocer los significados específicos del producto.

SQLSTATE

SQLSTATE también se establece por el gestor de bases de datos después de la ejecución de cada sentencia de SQL. Por lo tanto, los programas de aplicación pueden comprobar la ejecución de las sentencias de SQL probando SQLSTATE en lugar de SQLCODE.

SQLSTATE proporciona a los programas de aplicación códigos comunes para las condiciones de errores comunes. Además, SQLSTATE está diseñado para que los programas de aplicación puedan probar los errores específicos o las clases de error. El esquema de codificación es el mismo para todos los gestores de bases de datos IBM y se basa en el estándar SQL92 ISO/ANSI. Para obtener una lista completa de los valores posibles de SQLSTATE, consulte la publicación *Consulta de mensajes*.

comentarios de SQL

Las sentencias de SQL estáticas pueden incluir comentarios SQL o del lenguaje principal. Los comentarios SQL se introducen mediante dos guiones.

Estas reglas se aplican a la utilización de los comentarios SQL:

- Los dos guiones deben estar en la misma línea, no separados por un espacio.
- Los comentarios pueden empezar donde sea válido un espacio (excepto dentro de un símbolo delimitador o entre 'EXEC' y 'SQL').
- Los comentarios terminan al final de la línea.
- Los comentarios no están permitidos dentro de sentencias que están preparadas dinámicamente (utilizando PREPARE o EXECUTE IMMEDIATE).
- En COBOL, los guiones deben ir precedidos por un espacio.

Ejemplo: Este ejemplo muestra cómo incluir comentarios en una sentencia de SQL de un programa en C:

```
EXEC SQL
CREATE VIEW PRJ_MAXPER      -- proyectos con más personal de soporte
AS SELECT PROJNO, PROJNAME -- número y nombre del proyecto
FROM PROJECT
WHEREDEPTNO = 'E21'      -- código dept soporte de sistemas
AND PRSTAFF > 1;
```


ALTER BUFFERPOOL

especificado ningún tamaño en la *cláusula-except-on-nodes* de la sentencia CREATE de la agrupación de almacenamientos intermedios).

SIZE *número-de-páginas*

El tamaño de agrupación de almacenamientos intermedios especificado como el número de páginas.⁵⁶

EXTENDED STORAGE

Si se activa la configuración de almacenamiento ampliado⁵⁷, las páginas que se están migrando desde esta agrupación de almacenamientos intermedios se almacenarán en la antememoria del almacenamiento ampliado.

NOT EXTENDED STORAGE

Incluso si se activa la configuración del almacenamiento ampliado, las páginas que migran fuera de esta agrupación de almacenamientos intermedios, NO se pondrán en antememoria en el almacenamiento intermedio.

ADD NODEGROUP *nombre-gruponodos*

Añade este grupo de nodos a la lista de grupos de nodos a la que se puede aplicar la definición de agrupación de almacenamientos intermedios. Para cualquier partición del grupo de nodos que no tenga ya definida la agrupación de almacenamientos intermedios, ésta se crea en la partición utilizando el tamaño por omisión especificado para la agrupación de almacenamientos intermedios. Los espacios de tablas del *nombre-gruponodos* pueden especificar esta agrupación de almacenamientos intermedios. El grupo de nodos debe existir actualmente en la base de datos (SQLSTATE 42704).

Notas

- Aunque la definición de agrupación de almacenamientos intermedios es transaccional y los cambios en la definición de la agrupación de almacenamientos intermedios se reflejará en las tablas del catálogo en la confirmación, los cambios no surtirán efecto en la agrupación de almacenamientos intermedios hasta la próxima vez que se inicie la base de datos. Los atributos actuales de la agrupación de almacenamientos intermedios existirán hasta entonces y no afectará a la agrupación de almacenamientos intermedios mientras tanto. Las tablas creadas en espacios de tablas de grupos de nodos nuevos utilizarán la agrupación de almacenamientos intermedios por omisión.

56. El tamaño puede especificarse con un valor de (-1), que indicará que el tamaño de la agrupación de almacenamientos intermedios deberá obtenerse del parámetro de configuración de base de datos BUFPAGE.

57. La configuración del almacenamiento ampliado se activa estableciendo los parámetros de configuración de la base de datos NUM_ESTORE_SEGS y ESTORE_SEG_SIZE en valores que no sean cero. Consulte la publicación *Administration Guide* para obtener detalles.

- Debe haber suficiente memoria real en la máquina para el total de las agrupaciones de almacenamientos intermedios, así como para el resto de necesidades de espacio del gestor de bases de datos y de las aplicaciones.

ALTER NICKNAME

ALTER NICKNAME

La sentencia ALTER NICKNAME modifica la representación que la base de datos federada hace de la tabla o la vista de la fuente de datos:

- Cambiando los nombres locales de las columnas de la tabla o la vista
- Cambiando los tipos de datos locales de estas columnas
- Añadiendo, cambiando o suprimiendo opciones para estas columnas

Invocación

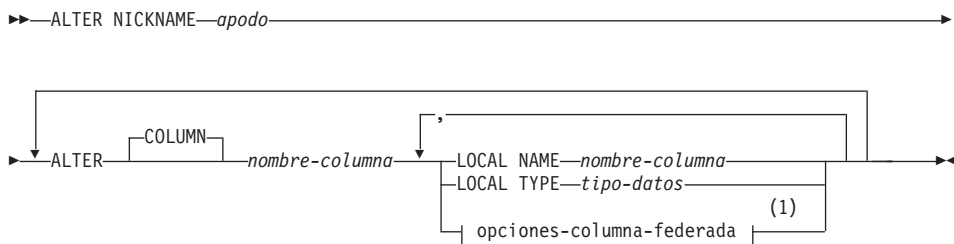
Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que se puede preparar dinámicamente. Sin embargo, si la opción de enlace DYNAMICRULES BIND es aplicable, la sentencia no se puede preparar dinámicamente (SQLSTATE 42509).

Autorización

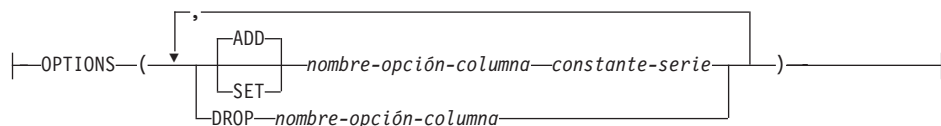
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Privilegio ALTER para el apodo especificado en la sentencia
- Privilegio CONTROL para el apodo especificado en la sentencia
- Privilegio ALTERIN para el esquema, si existe el nombre de esquema del apodo
- El definidor del apodo tal como está registrado en la columna DEFINER de la vista de catálogo para el apodo

Sintaxis



opciones-columna-federada:



Notas:

- 1 Si el usuario necesita especificar la cláusula opciones-columna-federada además del parámetro LOCAL NAME, el parámetro LOCAL TYPE o ambos, debe especificar la cláusula opciones-columna-federada en último lugar.

Descripción*apodo*

Identifica el apodo para la tabla o la vista de la fuente de datos que contiene la columna especificada después de la palabra clave COLUMN. Debe ser un apodo descrito en el catálogo.

ALTER COLUMN *nombre-columna*

Los nombres de la columna que se han de modificar. El *nombre-columna* es el nombre actual del servidor federado para la columna de la tabla o la vista de la fuente de datos. El *nombre-columna* debe identificar una columna existente de la tabla o la vista de la fuente de datos a la que el *apodo* hace referencia.

LOCAL NAME *nombre-columna*

Es el nuevo nombre por el cual el servidor federado va a hacer referencia a la columna identificada por el parámetro ALTER COLUMN *nombre-columna*. Este nuevo nombre debe ser un identificador DB2 válido.

LOCAL TYPE *tipo-datos*

Correlaciona el tipo de datos de la columna especificada con un tipo de datos local distinto del que se correlaciona ahora. El nuevo tipo se indica por *tipo-datos*.

El *tipo-datos* no puede ser LONG VARCHAR, LONG VARGRAPHIC, DATALINK, un tipo de datos de objeto grande (LOB) ni un tipo definido por el usuario.

OPTIONS

Indica las opciones de columna que se han de habilitar, restaurar o desactivar para la columna especificada después de la palabra clave COLUMN. Consulte el apartado "Opciones de columna" en la página 1413 para obtener las descripciones de *nombres-opción-columna* y sus valores.

ADD

Habilita una opción de columna.

SET

Cambia el valor de una opción de columna.

nombre-opción-columna

Nombra una opción de columna que se ha de habilitar o restaurar.

ALTER NICKNAME

constante-serie

Especifica el valor para *nombre-opción-columna* como una constante de serie de caracteres.

DROP *nombre-opción-columna*

Desactiva una opción de columna.

Normas

- Si una vista se ha creado en un apodo, no se puede utilizar la sentencia ALTER NICKNAME para cambiar los nombres locales ni los tipos de datos para las columnas de la tabla o la vista a la que el apodo hace referencia (SQLSTATE 42601). Sin embargo, la sentencia se puede utilizar para habilitar, restaurar o desactivar las opciones de columna para estas columnas.

Notas

- Si se utiliza ALTER NICKNAME para cambiar el nombre local de una columna de una tabla o vista a la que un apodo hace referencia, las consultas de la columna deben hacerle referencia por su nuevo nombre.
- No se puede especificar una opción de columna más de una vez en la misma sentencia ALTER NICKNAME (SQLSTATE 42853). Cuando se habilita, restaura o desactiva una opción de columna, no afecta a ninguna otra opción de columna que se esté utilizando.
- Cuando se cambia la especificación local del tipo de datos de una columna, el gestor de bases de datos invalida cualquier estadística (HIGH2KEY, LOW2KEY, etcétera) reunida para esa columna.
- La sentencia ALTER NICKNAME no se puede procesar en una unidad de trabajo (UOW) si una sentencia SELECT de la misma UOW ya hace referencia al apodo al que esta sentencia hace referencia (SQLSTATE 55007).

Ejemplos

Ejemplo 1: El apodo NICK1 hace referencia a una tabla DB2 Universal Database para AS/400 llamada T1. También, COL1 es el nombre local que hace referencia a la primera columna de esta tabla, C1. Cambie el nombre local C1 por NEWCOL.

```
ALTER NICKNAME NICK1
ALTER COLUMN COL1
LOCAL NAME NEWCOL
```

Ejemplo 2: El apodo EMPLOYEE hace referencia a una tabla DB2 Universal Database para OS/390 llamada EMP. También, SALARY es el nombre local que hace referencia a EMP_SAL, una de las columnas de esta tabla. El tipo de datos de la columna, FLOAT, se correlaciona con el tipo de datos local, DOUBLE. Cambie la correlación para que FLOAT se correlacione con DECIMAL (10, 5).

```
ALTER NICKNAME EMPLOYEE  
ALTER COLUMN SALARY  
LOCAL TYPE DECIMAL(10,5)
```

Ejemplo 3: Indique que en una tabla Oracle, una columna con el tipo de datos de VARCHAR no tiene blancos de cola. El apodo para la tabla es NICK2 y el nombre local para la columna es COL1.

```
ALTER NICKNAME NICK2  
ALTER COLUMN COL1  
OPTIONS ( ADD VARCHAR_NO_TRAILING_BLANKS 'Y' )
```

ALTER NODEGROUP

ALTER NODEGROUP

La sentencia ALTER NODEGROUP se utiliza para:

- añadir una o mas particiones o nodos a un grupo de nodos
- eliminar una o más particiones de un grupo de nodos.

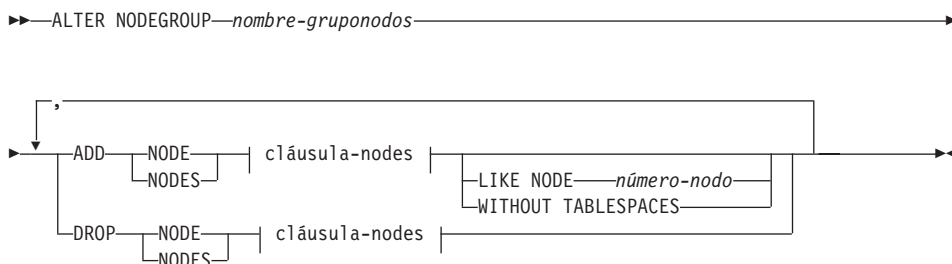
Invocación

Esta sentencia puede incluirse en un programa de aplicación o emitirse de forma interactiva. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

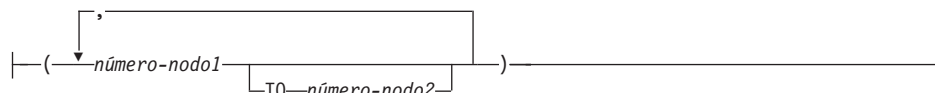
Autorización

El ID de autorización de la sentencia debe tener autorización SYSCTRL o SYSADM.

Sintaxis



cláusula-nodes:



Descripción

nombre-gruponodos

Indica el nombre del grupo de nodos. Este nombre sólo se compone de una parte. Se trata de un identificador de SQL (ordinario o bien delimitado). Debe ser un grupo de nodos descrito en el catálogo. No se puede especificar IBMCATGROUP ni IBMTEMPGROUP (SQLSTATE 42832).

ADD NODE

Especifica la partición o particiones específicas que se han de añadir al

grupo de nodos. NODES es un sinónimo de NODE. Cualquier partición especificada aún no debe estar definida en el grupo de nodos (SQLSTATE 42728).

DROP NODE

Especifica la partición o particiones específicas que se han de eliminar del grupo de nodos. NODES es un sinónimo de NODE. Cualquier partición especificada ya debe estar definida en el grupo de nodos (SQLSTATE 42729).

cláusula-nodes

Especifica la partición o particiones que se han de añadir o eliminar.

número-nodo1

Especifique un número de partición específico.

TO *número-nodo2*

Especifique un rango de números de partición. El valor de *número-nodo2* debe ser mayor o igual que el valor de *número-nodo1* (SQLSTATE 428A9).

LIKE NODE *número-nodo*

Especifica que los contenedores para los espacios de tablas existentes en el grupo de nodos serán los mismos que los contenedores especificados en el *número-nodo*. La partición especificada debe ser una partición que existía en el grupo de nodos antes de esta sentencia y que no está incluida en una cláusula DROP NODE de la misma sentencia.

WITHOUT TABLESPACES

Especifica que los espacios de la tabla por omisión no se crean en la partición o particiones que se acaban de añadir. La sentencia ALTER TABLESPACE que utiliza la cláusula FOR NODE debe utilizarse para definir los contenedores que se han de utilizar con los espacios de tablas que se definen en este grupo de nodos. Si no se especifica esta opción, se especifican los contenedores por omisión en las particiones para cada espacio de tablas definido en el grupo de nodos.

Normas

- Cada partición o nodo especificado por un número debe estar definido en el archivo `db2nodes.cfg` (SQLSTATE 42729). Vea “Partición de datos entre múltiples particiones” en la página 40 para obtener información sobre este archivo.
- Cada *número-nodo* listado en la cláusula ON NODES debe ser para una partición exclusiva (SQLSTATE 42728).
- Un número de partición válido está comprendido entre 0 y 999 inclusive (SQLSTATE 42729).
- Una partición no puede aparecer en las cláusulas ADD y DROP al mismo tiempo (SQLSTATE 42728).

ALTER NODEGROUP

- Debe quedar una partición como mínimo en el grupo de nodos. La última partición no se puede eliminar de un grupo de nodos (SQLSTATE 428C0).
- Si no se especifica la cláusula LIKE NODE ni la cláusula WITHOUT TABLESPACES cuando se añade una partición, la opción por omisión es utilizar el número de partición más bajo de las particiones existentes en el grupo de nodos (por ejemplo, 2) y continuar como si se hubiera especificado LIKE NODE 2. Para utilizar una partición existente como partición por omisión, debe tener contenedores definidos para todos los espacios de tablas del grupo de nodos (la columna IN_USE de SYSCAT.NODEGROUPDEF no es 'T').

Notas

- Cuando se añade una partición o nodo a un grupo de nodos, se crea una entrada de catálogo para la partición (vea SYSCAT.NODEGROUPDEF). El mapa de particionamiento se cambia inmediatamente para que incluya la nueva partición junto con un indicador (IN_USE) de que la partición está en el mapa de particionamiento si se cumple una de estas condiciones:
 - no hay espacios de tablas definidos en el grupo de nodos o
 - no hay tablas definidas en los espacios de tablas definidos en el grupo de nodos y no se ha especificado la cláusula WITHOUT TABLESPACES.

El mapa de particionamiento no se cambia y se habilita el indicador (IN_USE) para indicar que la partición no se incluye en el mapa de particionamiento si:

- existen tablas en los espacios de tablas del grupo de nodos o
- existen espacios de tablas en el grupo de nodos y se ha especificado la cláusula WITHOUT TABLESPACES.

Para cambiar el mapa de particionamiento, debe utilizarse el mandato REDISTRIBUTE NODEGROUP. Esto redistribuye los datos, cambia el mapa de particionamiento y cambia el indicador. Si se ha especificado la cláusula WITHOUT TABLESPACES, los contenedores de espacios de tabla se han de añadir antes de intentar redistribuir los datos.

- Cuando se elimina una partición de un grupo de nodos, se actualiza la entrada de catálogo creada para la partición (vea SYSCAT.NODEGROUPDEF). Si no hay tablas definidas en los espacios de tablas definidos en el grupo de nodos, se modifica inmediatamente el mapa de particionamiento para excluir la partición eliminada y se suprime la entrada de la partición en el grupo de nodos. Si hay tablas, el mapa de particionamiento no se modifica y se habilita el indicador (IN_USE) para indicar que la partición está esperando su eliminación. Debe utilizarse el mandato REDISTRIBUTE NODEGROUP para redistribuir los datos y eliminar la entrada de la partición en el grupo de nodos.

Ejemplo

Suponga que tiene una base de datos de seis particiones que presenta las particiones siguientes: 0, 1, 2, 5, 7 y 8. Se añaden dos particiones al sistema con los números de partición 3 y 6.

- Suponga que desea añadir ambas particiones o nodos 3 y 6 a un grupo de nodos llamado MAXGROUP y que tiene contenedores de espacios de tablas como los de la partición 2. La sentencia es la siguiente:

```
ALTER NODEGROUP MAXGROUP  
ADD NODES (3,6) LIKE NODE 2
```

- Suponga que desea eliminar la partición 1 y añadir la partición 6 al grupo de nodos MEDGROUP. Definirá los contenedores de espacios de tablas por separado para la partición 6 utilizando ALTER TABLESPACE. La sentencia es la siguiente:

```
ALTER NODEGROUP MEDGROUP  
ADD NODE(6) WITHOUT TABLESPACES  
DROP NODE(1)
```

ALTER SEQUENCE

ALTER SEQUENCE

La sentencia ALTER SEQUENCE puede utilizarse para cambiar una secuencia en cualquiera de estos modos:

- Reiniciando la secuencia
- Cambiando el incremento entre valores de secuencia futuros
- Estableciendo o eliminando los valores mínimo y máximo
- Cambiando los números de secuencia almacenados en antememoria
- Cambiando el atributo que determina si la secuencia puede realizar un ciclo o no
- Cambiando la posibilidad de que los números de secuencia deban generarse en orden de petición

Invocación

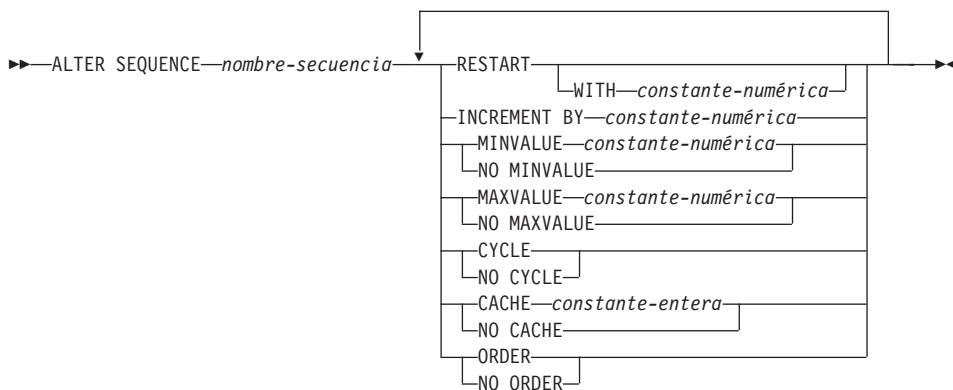
Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse dinámicamente. Sin embargo, si se aplica la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

Autorización

El ID de autorización de la sentencia debe tener al menos uno de los siguientes privilegios:

- Definidor de la secuencia
- Privilegio ALTERIN para el esquema especificado implícita o explícitamente
- Autorización SYSADM o DBADM

Sintaxis



Descripción

nombre-secuencia

Identifica la secuencia en concreto que se debe cambiar. El nombre, incluido el calificador de esquema implícito o explícito, debe identificar de forma exclusiva una secuencia existente en el servidor actual. Si no existe ninguna secuencia con este nombre en el esquema especificado explícita o implícitamente, se emitirá un error (SQLSTATE 42704). *nombre-secuencia* no debe ser una secuencia generada por el sistema para una columna de identidad (SQLSTATE 428FB).

RESTART

Reinicia la secuencia. Si no se especifica *constante-numérica*, la secuencia se reinicia en el valor especificado implícita o explícitamente como el valor inicial en la sentencia CREATE SEQUENCE que ha creado originalmente la secuencia.

WITH *constante-numérica*

Reinicia la secuencia con el valor especificado. Este valor puede ser cualquier valor positivo o negativo que pueda asignarse a una columna del tipo de datos asociado con la secuencia (SQLSTATE 42820), a condición de que no haya dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA).

INCREMENT BY

Especifica el intervalo entre valores consecutivos de la secuencia. Este valor puede ser cualquier valor positivo o negativo que pueda asignarse a una columna del tipo de datos asociado con la secuencia (SQLSTATE 42820), que no exceda el valor de una constante entera grande (SQLSTATE 42815) y que no tenga dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA).

Si este valor es negativo, se trata de una secuencia descendente. Si este valor es 0 o el valor absoluto es mayor que MAXVALUE - MINVALUE o positivo, se trata de una secuencia ascendente.

MINVALUE o NO MINVALUE

Especifica el valor mínimo en el que una secuencia descendente pasa por un ciclo o deja de generar valores o en el que una secuencia ascendente pasa por un ciclo después de alcanzar el valor máximo.

MINVALUE *constante-numérica*

Especifica la constante numérica que es el valor mínimo. Este valor puede ser cualquier valor positivo o negativo que pueda asignarse a una columna del tipo de datos asociado con la secuencia (SQLSTATE 42820), que no tenga dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA) pero que sea inferior o igual al valor máximo (SQLSTATE 42815).

ALTER SEQUENCE

NO MINVALUE

Para una secuencia ascendente, el valor es el valor de inicio original.
Para una secuencia descendente, el valor es el valor mínimo del tipo de datos asociado con la secuencia.

MAXVALUE o NO MAXVALUE

Especifica el valor máximo en el que una secuencia ascendente pasa por un ciclo o deja de generar valores o en el que una secuencia descendente pasa por un ciclo después de alcanzar el valor mínimo.

MAXVALUE *constante-numérica*

Especifica la constante numérica que es el valor máximo. Este valor puede ser cualquier valor positivo o negativo que pueda asignarse a una columna del tipo de datos asociado con la secuencia (SQLSTATE 42820), que no tenga dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA) pero que sea superior o igual al valor mínimo (SQLSTATE 42815).

NO MAXVALUE

Para una secuencia ascendente, el valor es el valor máximo del tipo de datos asociado con la secuencia. Para una secuencia descendente, el valor es el valor de inicio original.

CYCLE o NO CYCLE

Especifica si la secuencia debe continuar generando valores después de alcanzar el valor máximo o el valor mínimo. El límite de la secuencia puede alcanzarse con el siguiente valor que coincida exactamente con la condición de límite o excediendo el valor.

CYCLE

Especifica que se continúan generando valores para esta secuencia después de haber alcanzado el valor máximo o mínimo. Si se utiliza esta opción, cuando una secuencia ascendente haya alcanzado su valor máximo, generará su valor mínimo; o cuando una secuencia descendente haya alcanzado su valor mínimo, generará su valor máximo. Los valores máximo y mínimo para la secuencia determinan el rango que se utiliza para el ciclo.

Cuando CYCLE está en vigor, DB2 puede generar valores duplicados para la secuencia.

NO CYCLE

Especifica que no se generarán valores para la secuencia una vez que se haya alcanzado el valor máximo o mínimo para la secuencia.

CACHE o NO CACHE

Especifica si se deben mantener algunos valores preasignados en memoria para obtener un acceso más rápido. Esta opción se utiliza para el rendimiento y el ajuste.

CACHE *constante-entera*

Especifica el número máximo de valores de secuencia que se preasignan y se mantienen en memoria. La preasignación y el almacenamiento de valores en la antememoria reduce la E/S síncrona en las anotaciones cronológicas cuando se generan valores para la secuencia.

En el caso de producirse una anomalía del sistema, todos los valores de secuencia almacenados en antememoria que no se han utilizado en sentencias confirmadas se pierden (es decir, no se utilizarán nunca). El valor especificado para la opción **CACHE** es el número máximo de valores de secuencia que puede perderse en caso de anomalía del sistema.

El valor mínimo es 2 (SQLSTATE 42815).

NO CACHE

Especifica que los valores de la secuencia no se deben preasignar. Asegura que no haya ninguna pérdida de valores en el caso de producirse una anomalía del sistema, un cierre o una desactivación de la base de datos. Cuando se especifica esta opción, los valores de la secuencia no se almacenan en la antememoria. En este caso, cada petición de un valor nuevo para la secuencia produce E/S síncrona en las anotaciones cronológicas.

NO ORDER o ORDER

Especifica si los números de secuencia deben generarse según el orden de petición.

ORDER

Especifica que los números de secuencia se generan según el orden de petición.

NO ORDER

Especifica que los números de secuencia no necesitan generarse según el orden de petición.

Después de reiniciar una secuencia o de cambiar a **CYCLE**, es posible que los números de secuencia sean valores duplicados de los números generados por la secuencia anteriormente.

Notas

- Sólo los números de secuencia futuros se ven afectados por la secuencia **ALTER SEQUENCE**.
- El tipo de datos de una secuencia no se puede cambiar. En lugar de ello, elimine la secuencia y vuelva a crear otra especificando el tipo de datos deseado para la secuencia nueva.
- Todos los valores almacenados en antememoria se pierden cuando se modifica una secuencia.

ALTER SEQUENCE

- También se soporta la sintaxis siguiente: NOMINVALUE, NOMAXVALUE, NOCYCLE, NOCACHE y NOORDER.

Ejemplos

Ejemplo 1: Una posible razón para especificar RESTART sin un valor numérico puede ser para restablecer la secuencia en el valor START WITH. En este ejemplo, el objetivo es generar los números del 1 hasta el número de filas de tabla y, a continuación, insertar los números en una columna añadida a la tabla utilizando tablas temporales. También se podría utilizar para obtener resultados en los que todas las filas resultantes estén numeradas:

```
ALTER SEQUENCE ORG_SEQ RESTART  
SELECT NEXTVAL FOR ORG_SEQ, ORG.* FROM ORG
```

ALTER SERVER

La sentencia ALTER SERVER ⁵⁸ se utiliza para:

- Modificar la definición de una fuente de datos específica o la definición de una categoría de fuentes de datos
- Realizar cambios en la configuración de una fuente de datos específica o en la configuración de una categoría de fuentes de datos—cambios que persisten a través de múltiples conexiones a la base de datos federada.

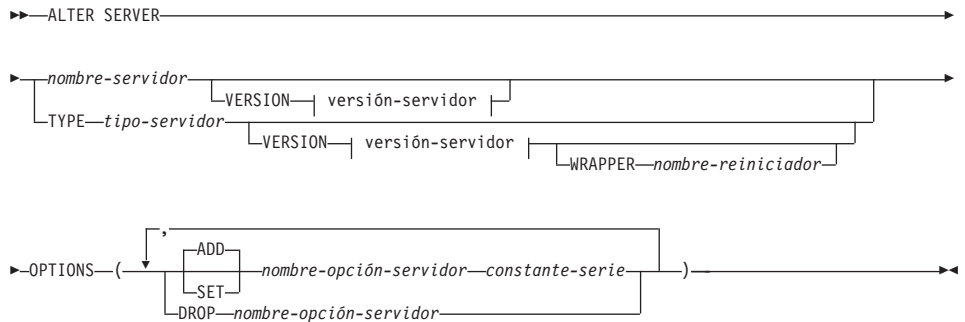
Invocación

Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si se aplica la opción de enlace lógico DYNAMICRULES BIND, la sentencia no puede prepararse dinámicamente (SQLSTATE 42509).

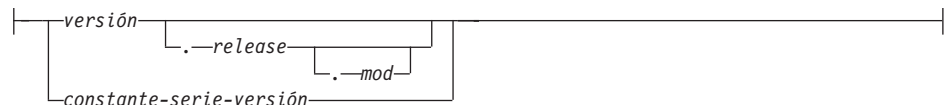
Autorización

El ID de autorización de la sentencia debe incluir la autorización SYSADM o DBADM en la base de datos federada.

Sintaxis



versión-servidor:



58. En esta sentencia, la palabra SERVER y los nombres de parámetros que terminan en *-servidor* sólo hacen referencia a las fuentes de datos de un sistema federado. No hacen referencia al servidor federado de dicho sistema ni a los servidores de aplicaciones DRDA. Para obtener información acerca de los sistemas federados, consulte “Sistemas federados DB2” en la página 56. Para obtener información acerca de los servidores de aplicaciones DRDA, consulte “Base de datos relacional distribuida” en la página 43.

ALTER SERVER

Descripción

nombre-servidor

Identifica el nombre del servidor federado para la fuente de datos a la que se han de aplicar los cambios que se están solicitando. La fuente de datos debe ser una descrita en el catálogo.

VERSION

Después de *nombre-servidor*, VERSION y su parámetro especifican una nueva versión de la fuente de datos que indica *nombre-servidor*.

versión

Especifica el número de versión. *versión* debe ser un entero.

release

Especifica el número de release de la versión indicada por *versión*. *release* debe ser un entero.

mod

Especifica el número de la modificación del release indicado por *release*. *mod* debe ser un entero.

constante-serie-versión

Especifica la designación completa de la versión. La *constante-serie-versión* puede ser un solo valor (por ejemplo, '8i'); o puede ser los valores concatenados de *versión*, *release* y, si se aplica, *mod* (por ejemplo, '8.0.3').

TYPE *tipo-servidor*

Especifica el tipo de fuente de datos al que se han de aplicar los cambios que se están solicitando. El tipo de servidor debe ser uno listado en el catálogo.

VERSION

Después de *tipo-servidor*, VERSION y su parámetro especifican la versión de las fuentes de datos para las cuales se han de habilitar, restaurar o desactivar las opciones de servidor.

WRAPPER *nombre-reiniciador*

Especifica el nombre del reiniciador que el servidor federado utiliza para interactuar con las fuentes de datos del tipo y versión indicados por *tipo-servidor* y *versión-servidor*. El reiniciador debe estar listado en el catálogo.

OPTIONS

Indica las opciones de servidor que se han de habilitar, restaurar o desactivar para la fuente de datos indicada por *nombre-servidor* o para la categoría de fuentes de datos indicada por *tipo-servidor* y sus parámetros asociados. Consulte "Opciones de servidor" en la página 1415 para ver las descripciones de *nombre-opción-servidor* y sus valores.

ADD

Habilita una opción de servidor.

SET

Cambia el valor de una opción de servidor.

nombre-opción-servidor

Nombra una opción de servidor que se ha de habilitar o restaurar.

constante-serie

Especifica un valor para *nombre-opción-servidor* como una constante de serie de caracteres.

DROP *nombre-opción-servidor*

Desactiva una opción de servidor.

Notas

- Esta sentencia no soporta las opciones de servidor DBNAME y NODE (SQLSTATE 428EE).
- Una opción de servidor no se puede especificar más de una vez en la misma sentencia ALTER SERVER (SQLSTATE 42853). Cuando se habilita, restaura o desactiva una opción de servidor, no afecta a ninguna otra opción de servidor que se esté utilizando.
- Una sentencia ALTER SERVER de una unidad de trabajo (UOW) determinada no se puede procesar bajo ninguna de las condiciones siguientes:
 - La sentencia hace referencia a una sola fuente de datos y la UOW ya incluye una sentencia SELECT que hace referencia a un apodo de una tabla o la vista contenida en esta fuente de datos (SQLSTATE 55007).
 - La sentencia hace referencia a una categoría de una fuente de datos (por ejemplo, todas las fuentes de datos para un tipo específico y versión) y la UOW ya incluye una sentencia SELELCT que hace referencia a un apodo de una tabla o vista contenida en una de estas fuentes de datos (SQLSTATE 55007).
- Si la opción de servidor se establece en un valor para un tipo de fuente de datos y se establece en otro valor para una instancia del tipo, el segundo valor prevalece sobre el primero para la instancia. Por ejemplo, suponga que PLAN_HINTS se establece en 'Y' para el tipo de servidor ORACLE y en 'N' para una fuente de datos Oracle llamada DELPHI. Esta configuración provoca que se habiliten las indicaciones de planes en todas las fuentes de datos Oracle excepto en DELPHI.

Ejemplos

Ejemplo 1: Asegúrese de que cuando se envían ID de autorización a las fuentes de datos Oracle 8.0.3, no se cambian las mayúsculas y minúsculas de los ID.

ALTER SERVER

También, suponga que estas fuentes de datos han iniciado su ejecución en una CPU actualizada que es la mitad de rápida que la CPU local. Informe al optimizador de esta estadística.

```
ALTER SERVER  
  TYPE ORACLE  
  VERSION 8.0.3  
  OPTIONS  
    ( ADD FOLD_ID 'N',  
      SET CPU_RATIO '2.0' )
```

Ejemplo 2: Indique que una fuente de datos DB2 Universal Database para AS/400 Versión 3.0 llamada SUNDIAL se ha actualizado a la Versión 3.1.

```
ALTER SERVER SUNDIAL  
  VERSION 3.1
```

ALTER TABLE

La sentencia ALTER TABLE modifica las tablas existentes al:

- Añadir una o más columnas a una tabla
- Añadir o eliminar una clave primaria
- Añadir o eliminar una o más restricciones de unicidad o de referencia
- Añadir o eliminar una o más definiciones de restricción de comprobación
- Alterar la longitud de una columna VARCHAR
- Alterar una columna de tipo de referencia para añadir un ámbito
- Alterar la expresión de generación de una columna generada
- Añadir o eliminar una clave de particionamiento
- Cambiar atributos de una tabla, tales como la opción DATA CAPTURE, PCTFREE, LOCKSIZE o la modalidad APPEND.
- Poner la tabla en un estado de NOT LOGGED INITIALLY.

Invocación

Esta sentencia se puede intercalar en un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámicas. Es una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar dinámicamente (SQLSTATE 42509).

Autorización

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Privilegio ALTER para la tabla que se debe modificar
- Privilegio CONTROL para la tabla que se debe modificar
- Privilegio ALTERIN para el esquema de la tabla
- Autorización SYSADM o DBADM.

Para crear o eliminar una clave foránea, el ID de autorización de la sentencia debe tener uno de los privilegios siguientes para la tabla padre:

- Privilegio REFERENCES para la tabla
- Privilegio REFERENCES para todas las columnas de la clave padre especificada
- Privilegio CONTROL para la tabla
- Autorización SYSADM o DBADM.

Para eliminar una clave primaria o una restricción de unicidad de la tabla T, el ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes para cada tabla que sea dependiente de esta clave padre T:

ALTER TABLE

- Privilegio ALTER para la tabla
- Privilegio CONTROL para la tabla
- Privilegio ALTERIN para el esquema de la tabla
- Autorización SYSADM o DBADM.

Para modificar una tabla y convertirla en una tabla de resumen (utilizando una selección completa), el ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Privilegio CONTROL para la tabla
- Autorización SYSADM o DBADM;

y como mínimo uno de los privilegios siguientes, para cada tabla o vista identificada en la selección completa:

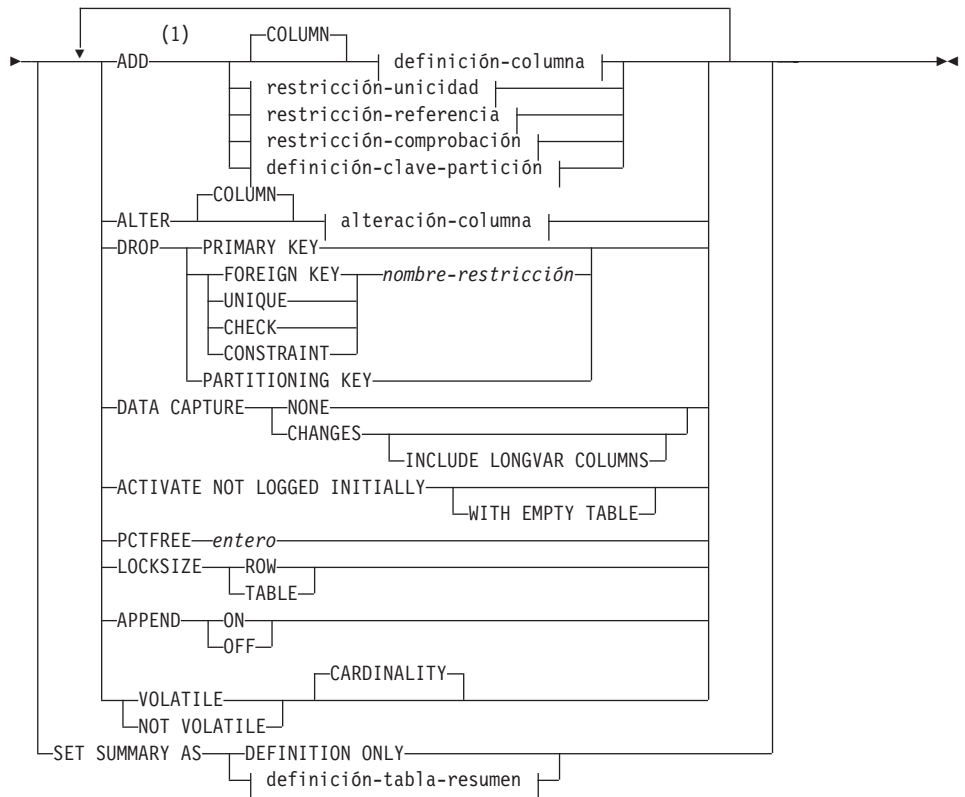
- Privilegio SELECT y ALTER para la tabla o vista
- Privilegio CONTROL para la tabla o vista
- Privilegio SELECT para la tabla o vista y privilegio ALTERIN para el esquema de la tabla o vista
- Autorización SYSADM o DBADM.

Para convertir una tabla de resumen en una tabla regular, el ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes, para cada tabla o vista identificada en la selección completa utilizada para definir la tabla de resumen:

- Privilegio ALTER para la tabla o vista
- Privilegio CONTROL para la tabla o vista
- Privilegio ALTERIN para el esquema de la tabla o vista
- Autorización SYSADM o DBADM

Sintaxis

►—ALTER TABLE—*nombre-tabla*—►

**Notas:**

- Para que haya compatibilidad con la Versión 1, la palabra clave ADD es opcional para:
 - las restricciones PRIMARY KEY sin nombre
 - las restricciones de referencia sin nombre
 - las restricciones de referencia cuyo nombre vaya a continuación de la frase FOREIGN KEY.

definición-tabla-resumen:

| (—selección completa—) | opciones-tabla-renovables |

ALTER TABLE

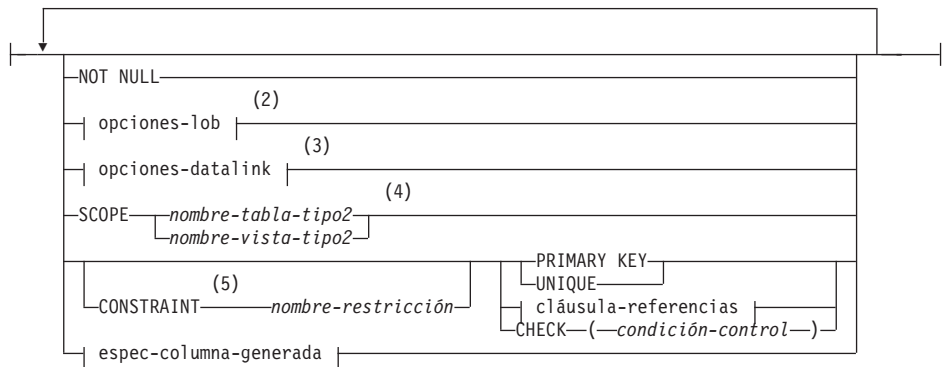
opciones-tabla-renovables:



definición-columna:



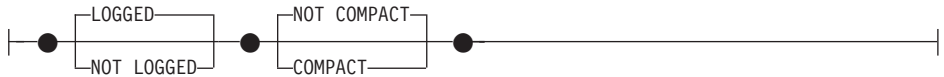
opciones-columna:



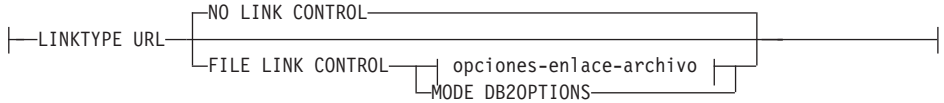
Notas:

- 1 Si la primera opción de columna elegida es la especificación de columna generada, se puede omitir el tipo de datos y ser calculado por la expresión de generación.
- 2 La cláusula opciones-lob sólo se aplica a tipos de gran objeto (BLOB, CLOB y DBCLOB) y a los tipos diferenciados basados en tipos de gran objeto.
- 3 La cláusula opciones-datalink sólo se aplica al tipo DATALINK y a los tipos diferenciados basados en el tipo DATALINK.
- 4 La cláusula SCOPE sólo se aplica al tipo REF.
- 5 Para que exista compatibilidad con la Versión 1, se puede omitir la palabra clave CONSTRAINT en una *definición-columna* que defina una cláusula-referencia.

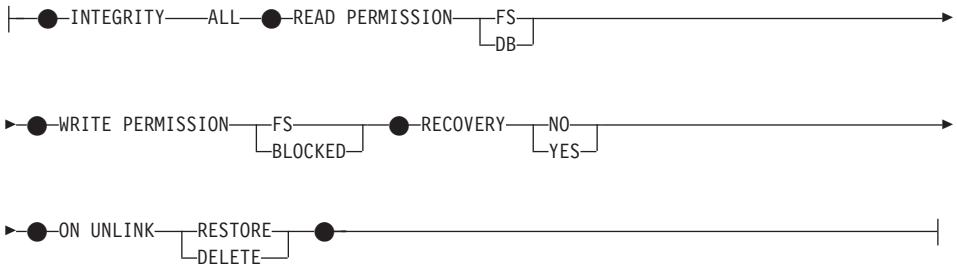
opciones-lob:



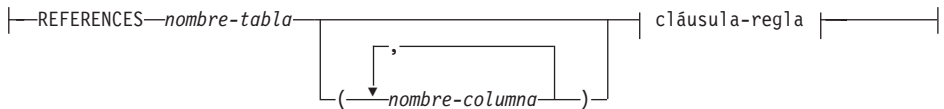
opciones-datalink:



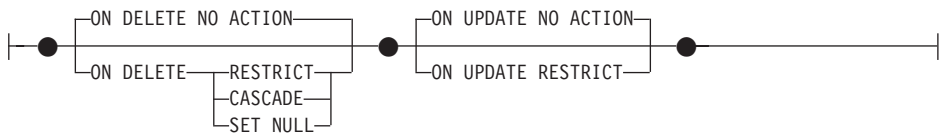
opciones-enlace-archivo:



cláusula-referencias:



cláusula-regla:



espec-columna-generada:

ALTER TABLE

cláusula-predefinida
GENERATED ALWAYS AS (—expresión-generación—)

cláusula-predefinida:

WITH DEFAULT
—constante—
—registro-especial-fechahora—
—USER—
—NULL—
función-conversión (—constante—
—registro-especial-fechahora—
—USER—)

restricción-unicidad:

UNIQUE
PRIMARY KEY
—CONSTRAINT—nombre-restricción
(—nombre-columna—)

restricción-referencia:

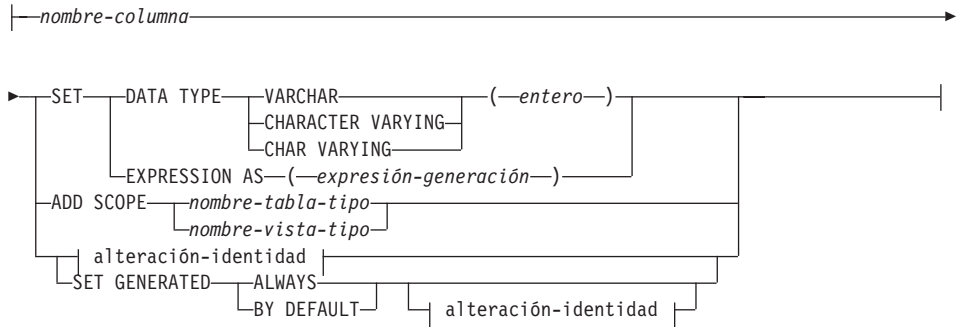
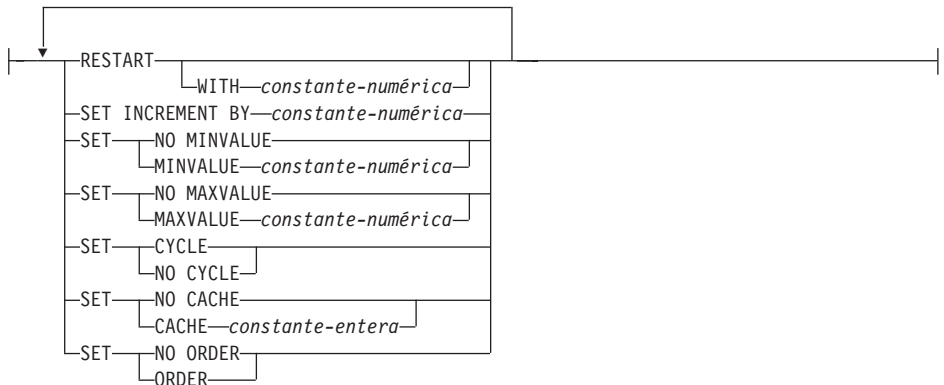
(1)
—CONSTRAINT—nombre-restricción
FOREIGN KEY (—nombre-columna—) cláusula-referencias

restricción-comprobación:

CHECK (—condición-control—)
—CONSTRAINT—nombre-restricción

definición-clave-partición:

PARTITIONING KEY (—nombre-columna—) USING HASHING

alteración-columna:**alteración-identidad:****Notas:**

- 1 Por razones de compatibilidad con la Versión 1, *nombre-restricción* se puede especificar a continuación de FOREIGN KEY (sin la palabra clave CONSTRAINT).

Descripción*nombre-tabla*

Identifica la tabla que se va a cambiar. Debe ser una tabla descrita en el catálogo y no debe ser una vista ni una tabla del catálogo. Si *nombre-tabla* identifica una tabla de resumen, las alteraciones se limitan a establecer la tabla de resumen en DEFINITION ONLY, activar NOT LOGGED INITIALLY, cambiar PCTFREE, LOCKSIZE, APPEND o VOLATILE. El *nombre-tabla* no puede ser un apodo (SQLSTATE 42809) ni una tabla temporal declarada (SQLSTATE 42995).

ALTER TABLE

SET SUMMARY AS

Permite modificar las propiedades de una tabla de resumen.

DEFINITION ONLY

Cambia una tabla de resumen de forma de que deja de ser considerada como una tabla de resumen. La tabla especificada por *nombre-tabla* debe estar definida como tabla de resumen que no está duplicada (SQLSTATE 428EW). La definición de las columnas de *nombre-tabla* no se modifica, pero la tabla ya no puede utilizarse para la optimización de consultas y ya no puede utilizarse la sentencia REFRESH TABLE.

definición-tabla-resumen

Modifica una tabla regular para convertirla en una tabla de resumen y utilizarla en la optimización de consultas. La tabla especificada por *nombre-tabla*:

- No debe estar previamente definida como tabla de resumen
- No debe ser una tabla con tipo
- No debe tener ninguna restricción, índice de unicidad ni desencadenante definidos
- No debe estar referenciada en la definición de otra tabla de resumen.

Si *nombre-tabla* no cumple estos criterios, se devuelve un error (SQLSTATE 428EW).

selección completa

Define la consulta en la que se basa la tabla. Las columnas de la tabla existente:

- deben tener el mismo número de columnas
- deben tener exactamente los mismos tipos de datos
- deben tener los mismos nombres de columna en las mismas posiciones de orden

que las columnas resultantes de la *selección completa* (SQLSTATE 428EW). Para obtener detalles sobre la especificación de la *selección completa* para una tabla de resumen, vea "CREATE TABLE" en la página 835. Una restricción adicional es que *nombre-tabla* no puede estar referenciada, directa o indirectamente, en la selección completa.

opciones-tabla-renovables

Especifica las opciones renovables para modificar una tabla de resumen.

DATA INITIALLY DEFERRED

Los datos de la tabla deben validarse utilizando la sentencia REFRESH TABLE o SET INTEGRITY.

REFRESH

Indica cómo se mantienen los datos de la tabla.

DEFERRED

Los datos de la tabla pueden renovarse en cualquier momento mediante la sentencia REFRESH TABLE. Los datos de la tabla sólo reflejan el resultado de la consulta en forma de instantánea del momento en que se procesa la sentencia REFRESH TABLE. Las tablas de resumen definidas con este atributo no permiten las sentencias INSERT, UPDATE o DELETE (SQLSTATE 42807).

IMMEDIATE

Los cambios realizados en las tablas principales como parte de una sentencia DELETE, INSERT o UPDATE se aplican en cascada a la tabla de resumen. En este caso, el contenido de la tabla, en cualquier momento, es igual como si se procesara la subselección especificada. Las tablas de resumen definidas con este atributo no permiten las sentencias INSERT, UPDATE o DELETE (SQLSTATE 42807).

ENABLE QUERY OPTIMIZATION

La tabla de resumen se puede utilizar para la optimización de consultas.

DISABLE QUERY OPTIMIZATION

La tabla de resumen no se utilizará para la optimización de consultas. La tabla todavía puede consultarse directamente.

ADD *definición-columna*

Añade una columna a la tabla. La tabla no debe ser una tabla con tipo (SQLSTATE 428DH). Si la tabla contiene filas, cada valor de la nueva columna añadida es su valor por omisión. La nueva columna es la última columna de la tabla. Es decir, si inicialmente existen n columnas, la columna añadida es la columna $n+1$. El valor den no puede ser mayor que 499.

La adición de la nueva columna no debe hacer que la cuenta total de bytes de todas las columnas exceda el tamaño máximo de registro especificado en la Tabla 36 en la página 1260. Vea “Notas” en la página 881 para obtener más información.

nombre-columna

Es el nombre de la columna que se va a añadir a la tabla. El nombre

ALTER TABLE

no puede estar calificado. No pueden utilizarse nombres de columna existentes en la tabla (SQLSTATE 42711).

tipo-datos

Es uno de los tipos de datos listado bajo “CREATE TABLE” en la página 835.

NOT NULL

Evita que la columna contenga valores nulos. También debe especificarse la *cláusula-predefinida* (SQLSTATE 42601).

opciones-lob

Especifica opciones para los tipos de datos LOB. Consulte *opciones-lob* en “CREATE TABLE” en la página 835.

opciones-datalink

Especifica opciones para los tipos de datos DATALINK. Consulte *opciones-datalink* en “CREATE TABLE” en la página 835.

SCOPE

Especifica un ámbito para una columna de tipo de referencia.

nombre-tabla-tipo2

El nombre de una tabla con tipo. El tipo de datos de *nombre-columna* debe ser REF(S), donde S es el tipo de *nombre-tabla-tipo2* (SQLSTATE 428DM). No se realiza ninguna comprobación del valor por omisión de *nombre-columna* para asegurarse de si realmente el valor hace referencia a una fila existente de *nombre-tabla-tipo2*.

nombre-vista-tipo2

El nombre de una vista con tipo. El tipo de datos de *nombre-columna* debe ser REF(S), donde S es el tipo de *nombre-vista-tipo2* (SQLSTATE 428DM). No se realiza ninguna comprobación del valor por omisión de *nombre-columna* para asegurarse de si realmente el valor hace referencia a una fila existente de *nombre-vista-tipo2*.

CONSTRAINT *nombre-restricción*

Indica el nombre de la restricción. Un *nombre-restricción* no debe identificar una restricción que ya se ha especificado dentro de la misma sentencia ALTER TABLE, o como el nombre de cualquier otra restricción existente en la tabla (SQLSTATE 42710).

Si el usuario no especifica el nombre de restricción, el sistema genera un identificador de 18 caracteres exclusivo entre los identificadores de las restricciones existentes definidas en la tabla⁵⁹.

59. El identificador está formado por “SQL” seguido de una secuencia de 15 caracteres numéricos, generados por una función basada en la indicación de la hora.

Cuando se utiliza con una restricción PRIMARY KEY o UNIQUE, el *nombre-restricción* puede utilizarse como el nombre de un índice que se ha creado para dar soporte a la restricción. Consulte el apartado “Notas” en la página 591 para ver los detalles sobre los nombres de índice asociados a restricciones de unicidad.

PRIMARY KEY

Proporciona un método abreviado para definir una clave primaria compuesta de una sola columna. Por lo tanto, si se especifica PRIMARY KEY en la definición de la columna C, el efecto es el mismo que si se especifica la cláusula PRIMARY KEY(C) como cláusula separada. La columna no puede contener valores nulos, de manera que también debe especificarse NOT NULL (SQLSTATE 42831).

Consulte el apartado PRIMARY KEY en la descripción de la *restricción-unicidad* más abajo.

UNIQUE

Proporciona un método abreviado de definir una clave de unicidad compuesta de una sola columna. Por lo tanto, si se especifica UNIQUE KEY en la definición de la columna C, el efecto es el mismo que si se especifica la cláusula UNIQUE(C) como cláusula separada.

Consulte el apartado UNIQUE en la descripción de la *restricción-unicidad* más abajo.

cláusula-referencias

Proporciona un método abreviado de definir una clave externa compuesta de una sola columna. Así, si se especifica una cláusula-referencias en la definición de la columna C, el efecto es el mismo que si se especificase esa cláusula-referencias como parte de una cláusula FOREIGN KEY en la que C fuera la única columna identificada.

Vea *cláusula-referencias* en “CREATE TABLE” en la página 835.

CHECK (*condición-control*)

Proporciona un método abreviado de definir una restricción de comprobación que se aplica a una sola columna. Vea *condición-control* en “CREATE TABLE” en la página 835.

generar-espec-columna

Vea “CREATE TABLE” en la página 835 para obtener detalles sobre la generación de columnas.

cláusula-predefinida

Especifica un valor por omisión para la columna.

WITH

Palabra clave opcional.

ALTER TABLE

DEFAULT

Proporciona un valor por omisión en el caso de que no se suministre ningún valor en INSERT o se especifique uno como DEFAULT en INSERT o UPDATE. Si no se especifica un valor por omisión específico a continuación de la palabra clave DEFAULT, el valor por omisión depende del tipo de datos de la columna, tal como se muestra en la Tabla 21. Si una columna está definida como DATALINK o tipo estructurado, no puede especificarse una cláusula por omisión.

Si se define una columna utilizando un tipo diferenciado, el valor por omisión de la columna es el valor por omisión del tipo de datos fuente convertido al tipo diferenciado.

Tabla 21. Valores por omisión (cuando no se especifica ningún valor)

Tipo de datos	Valor por omisión
Numérico	0
Serie de caracteres de longitud fija	Blancos
Serie de caracteres de longitud variable	Una serie de longitud 0
Serie gráfica de longitud fija	Blancos de doble byte
Serie gráfica de longitud variable	Una serie de longitud 0
Fecha	Para las filas existentes, fecha que corresponde al 1 de enero de 0001. Para las filas añadidas, la fecha actual.
Hora	Para filas existentes, una hora que corresponde a las 0 horas, 0 minutos y 0 segundos. Para filas añadidas, la hora actual.
Indicación de la hora	Para las filas existentes, una fecha que corresponde al 1 de Enero, 0001 y una hora que corresponde a las 0 horas, 0 minutos, 0 segundos y 0 microsegundos. Para filas añadidas, la indicación de la hora actual.
Serie binaria (blob)	Una serie de longitud 0

La omisión de DEFAULT en una *definición-columna* da como resultado la utilización del valor nulo como valor por omisión para la columna.

Los tipos específicos de los valores que pueden especificarse con la palabra clave DEFAULT son los siguientes.

constante

Especifica la constante como el valor por omisión para la columna. La constante especificada debe:

- representar un valor que pueda asignarse a la columna de acuerdo con las reglas de asignación descritas en el Capítulo 3
- no ser una constante de coma flotante a menos que la columna esté definida con un tipo de datos de coma flotante
- no tener dígitos diferentes de cero fuera de la escala del tipo de datos de la columna si la constante es decimal (por ejemplo, 1,234 no puede ser el valor por omisión para una columna DECIMAL(5,2))
- estar expresada con un máximo de 254 caracteres, incluyendo los caracteres de comillas, cualquier carácter prefijo como la X para una constante hexadecimal, los caracteres del nombre de función completamente calificado y paréntesis, cuando la constante es el argumento de una *función-conversión*.

registro-especial-fechahora

Especifica el valor del registro especial de indicación de fecha y hora (CURRENT DATE, CURRENT TIME o CURRENT TIMESTAMP) en el momento de INSERT o UPDATE como valor por omisión para la columna. El tipo de datos de la columna debe ser el tipo de datos que corresponde al registro especial especificado (por ejemplo, el tipo de datos debe ser DATE cuando se especifica CURRENT DATE). Para las filas existentes, el valor es la fecha, hora o indicación de la hora actuales en el momento de procesar la sentencia ALTER TABLE.

USER

Especifica el valor del registro especial USER en el momento de ejecutar INSERT o UPDATE, como el valor por omisión para la columna. Si se especifica USER, el tipo de datos de la columna debe ser una serie de caracteres con una longitud no inferior al atributo de longitud de USER. Para las filas existentes, el valor es el ID de autorización de la sentencia ALTER TABLE.

NULL

Especifica NULL como valor por omisión para la columna. Si se ha especificado NO NULL, DEFAULT NULL no debe especificarse en la misma definición de columna.

ALTER TABLE

función-conversión

Esta forma de valor por omisión sólo puede utilizarse con las columnas definidas como tipo de datos diferenciado, BLOB o de indicación de fecha y hora (DATE, TIME o TIMESTAMP). Para el tipo diferenciado, a excepción de los tipos diferenciados basados en tipos BLOB o de indicación de fecha y hora, el nombre de la función debe coincidir con el nombre del tipo diferenciado de la columna. Si está calificado con un nombre de esquema, debe ser el mismo que el nombre de esquema del tipo diferenciado. Si no está calificado, el nombre de esquema procedente de la resolución de la función debe ser el mismo que el nombre de esquema del tipo diferenciado. Para un tipo diferenciado basado en un tipo de indicación de fecha y hora, en el que el valor por omisión es una constante, debe utilizarse una función y el nombre de ésta debe coincidir con el nombre del tipo fuente del tipo diferenciado, con un nombre de esquema implícito o explícito de SYSIBM. Para las demás columnas de indicación de fecha y hora, también puede utilizarse la correspondiente función de fecha y hora. Para un BLOB o tipo diferenciado basado en BLOB, debe utilizarse una función, y el nombre de la función debe ser BLOB, junto con SYSIBM como nombre de esquema implícito o explícito.

constante

Especifica una constante como argumento. La constante debe cumplir las reglas de una constante para el tipo fuente del tipo diferenciado, o para el tipo de datos si no es un tipo diferenciado. Si la función-conversión es BLOB, la constante debe ser de tipo serie.

registro-especial-fechahora

Especifica CURRENT DATE, CURRENT TIME o CURRENT TIMESTAMP. El tipo fuente del tipo diferenciado de la columna debe ser el tipo de datos que corresponde al registro especial especificado.

USER

Especifica el registro especial USER. El tipo de datos del tipo fuente del tipo diferenciado de la columna debe ser el tipo de datos serie con una longitud mínima de 8 bytes. Si la función-conversión es BLOB, el atributo de longitud debe ser como mínimo 8 bytes.

Si el valor especificado no es válido, se producirá un error (SQLSTATE 42894).

ADD restricción-unicidad

Define una restricción de clave primaria o de unicidad. No puede añadirse una restricción de unicidad o de clave primaria a una tabla que sea una subtabla (SQLSTATE 429B3). Si la tabla es la supertabla del principio de la jerarquía, la restricción se aplica a la tabla y a todas sus subtablas.

CONSTRAINT nombre-restricción

Indica el nombre de la restricción de clave primaria o de unicidad. Para obtener más información, vea *nombre-restricción* en “CREATE TABLE” en la página 835.

UNIQUE (nombre_columna...)

Define una clave de unicidad compuesta por las columnas identificadas. Las columnas identificadas deben estar definidas como NOT NULL. Cada *nombre-columna* debe identificar una columna de la tabla y la misma columna no debe estar identificada más de una vez. El nombre no puede estar calificado. El número de columnas identificadas no debe exceder de 16 y la suma de sus longitudes almacenadas no debe exceder de 1024 (consulte el apartado “Cuentas de bytes” en la página 885 para conocer las longitudes almacenadas de columnas). La longitud de una columna individual cualquiera no debe exceder de 255 bytes. No puede utilizarse ningún LOB, LONG VARCHAR, LONG VARGRAPHIC, DATALINK, un tipo diferenciado de cualquiera de estos tipos ni una columna de tipo estructurado como parte de una clave de unicidad (aunque el atributo de longitud de la columna sea lo bastante pequeño como para caber dentro del límite de 255 bytes) (SQLSTATE 42962). El conjunto de columnas de la clave de unicidad no puede ser el mismo que el conjunto de columnas de la clave primaria o de otra clave de unicidad (SQLSTATE 01543).⁶⁰ Cualquier valor existente en el conjunto de columnas identificadas debe ser exclusivo (SQLSTATE 23515).

Se realiza una comprobación para determinar si un índice existente coincide con la definición de clave de unicidad (ignorando cualquier columna INCLUDE del índice). Una definición de índice coincide si identifica el mismo conjunto de columnas sin tener en cuenta el orden de las columnas ni las especificaciones de dirección (ASC/DESC). Si se encuentra una definición de índice coincidente, la descripción del índice se cambia para indicar que el sistema lo necesita y se cambia por de unicidad (después de asegurar la exclusividad) si no era un índice de unicidad. Si la tabla tiene más de un índice coincidente, se selecciona un índice de unicidad existente (la selección es arbitraria). Si no se encuentra ningún índice coincidente, se creará automáticamente un índice de unicidad para las columnas, tal como

60. Si LANGLEVEL es SQL92E o MIA, entonces se devuelve un error, SQLSTATE 42891.

ALTER TABLE

se describe en CREATE TABLE. Consulte el apartado “Notas” en la página 591 para ver los detalles sobre los nombres de índice asociados a restricciones de unicidad.

CLAVE PRIMARIA *...(nombre-columna,)*

Define una clave primaria formada por las columnas identificadas. Cada *nombre-columna* debe identificar una columna de la tabla, y la misma columna no debe identificarse más de una vez. El nombre no puede estar calificado. El número de columnas identificadas no debe exceder de 16 y la suma de sus longitudes almacenadas no debe exceder de 1024 (consulte el apartado “Cuentas de bytes” en la página 885 para conocer las longitudes almacenadas). La longitud de una columna individual cualquiera no debe exceder de 255 bytes. La tabla no debe tener una clave primaria y las columnas identificadas deben definirse como NOT NULL. No puede utilizarse ningún LOB, LONG VARCHAR, LONG VARGRAPHIC, DATALINK, un tipo diferenciado de cualquiera de estos tipos ni una columna de tipo estructurado como parte de una clave primaria (aunque el atributo de longitud de la columna sea lo bastante pequeño como para caber dentro del límite de 255 bytes) (SQLSTATE 42962). El conjunto de columnas de la clave primaria no puede ser el mismo que el conjunto de columnas de una clave de unicidad (SQLSTATE 01543).⁶⁰ Los valores existentes en el conjunto de columnas identificadas deben ser exclusivos (SQLSTATE 23515).

Se realiza una comprobación para determinar si un índice existente coincide con la definición de clave primaria (ignorando cualquier columna INCLUDE del índice). Una definición de índice coincide si identifica el mismo conjunto de columnas sin tener en cuenta el orden de las columnas ni las especificaciones de dirección (ASC/DESC). Si se encuentra una definición de índice coincidente, la descripción del índice se cambia para que indique que es el índice principal, tal como necesita el sistema, y se cambia al de unicidad (después de asegurar la exclusividad) si no era un índice de unicidad. Si la tabla tiene más de un índice coincidente, se selecciona un índice de unicidad existente (la selección es arbitraria). Si no se encuentra ningún índice coincidente, se creará automáticamente un índice de unicidad para las columnas, tal como se describe en CREATE TABLE. Consulte el apartado “Notas” en la página 591 para ver los detalles sobre los nombres de índice asociados a restricciones de unicidad.

En una tabla sólo se puede definir una clave primaria.

ADD *restricción-referencia*

Define una restricción de referencia. Vea *restricción-referencial* en “CREATE TABLE” en la página 835.

ADD restricción-comprobación

Define una restricción de comprobación. Vea *restricción-comprobación* en “CREATE TABLE” en la página 835.

ADD definición-clave-partición

Define una clave de particionamiento. La tabla debe estar definida en un espacio de tablas de un grupo de nodos de una sola partición y todavía no debe tener ninguna clave de particionamiento. Si ya existe una clave de particionamiento para la tabla, se debe suprimir la clave existente antes de añadir la nueva clave de particionamiento.

No puede añadirse una clave de particionamiento a una tabla que sea una subtabla (SQLSTATE 428DH).

PARTITIONING KEY (nombre-columna...)

Define una clave de particionamiento utilizando las columnas especificadas. Cada *nombre-columna* debe identificar una columna de la tabla, y la misma columna no debe identificarse más de una vez. El nombre no puede estar calificado. Una columna no se puede utilizar como parte de una clave de particionamiento si el tipo de datos de la columna es un LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB, DATALINK, un tipo diferenciado de cualquiera de estos tipos o un tipo estructurado.

USING HASHING

Especifica la utilización de la función de generación aleatoria como el método de partición para la distribución de datos. Es el único método de partición soportado.

ALTER alteración-columna

Altera las características de una columna.

nombre-columna

Es el nombre de la columna que se va a alterar en la tabla. El *nombre-columna* debe identificar una columna existente de la tabla (SQLSTATE 42703). El nombre no puede estar calificado.

SET DATA TYPE VARCHAR (entero)

Aumenta la longitud de una columna VARCHAR existente. CHARACTER VARYING o CHAR VARYING se pueden utilizar como sinónimos de la palabra clave VARCHAR. El tipo de datos de *nombre-columna* debe ser VARCHAR y la longitud máxima actual definida para la columna no debe ser superior al valor de *entero* (SQLSTATE 42837). El valor de *entero* puede ser de hasta 32672. La tabla no debe ser una tabla con tipo (SQLSTATE 428DH).

La alteración de la columna no debe hacer que la cuenta total de bytes de todas las columnas sobrepase el tamaño máximo de registro especificado en la Tabla 36 en la página 1260 (SQLSTATE 54010). Vea “Notas” en la página 881 para obtener más información. Si la columna

ALTER TABLE

se utiliza en un índice o restricción de unicidad, la nueva longitud no debe ser mayor que 255 bytes y no debe hacer que la suma de las longitudes almacenadas correspondientes al índice o restricción de unicidad sea mayor que 1024 (SQLSTATE 54008) (consulte el apartado “Cuentas de bytes” en la página 885 para conocer las longitudes almacenadas).

SET EXPRESSION AS (*expresión-generación*)

Cambia la expresión de la columna a la *expresión-generación* especificada. SET EXPRESSION AS requiere colocar la tabla en el estado de comprobación pendiente, utilizando la sentencia SET INTEGRITY. Después de la sentencia ALTER TABLE, se debe utilizar la sentencia SET INTEGRITY para actualizar y comparar todos los valores de la columna con la nueva expresión. La columna ya debe estar definida como columna generada basada en una expresión (SQLSTATE 42837). La expresión de generación debe seguir las mismas reglas que se aplican al definir una columna generada. El tipo de datos resultante de la expresión de generación se debe poder asignar al tipo de datos de la columna (SQLSTATE 42821).

ADD SCOPE

Añade un ámbito a una columna de tipo de referencia existente que todavía no tiene un ámbito definido (SQLSTATE 428DK). Si la tabla a alterar es una tabla con tipo, la columna no debe ser herencia de una supertabla (SQLSTATE 428DJ). Consulte “ALTER TYPE (Estructurado)” en la página 603 para obtener ejemplos.

nombre-tabla-tipo

El nombre de una tabla con tipo. El tipo de datos de *nombre-columna* debe ser REF(*S*), donde *S* es el tipo de *nombre-tabla-tipo* (SQLSTATE 428DM). No se realiza ninguna comprobación de los valores existentes de *nombre-columna* para asegurarse de si realmente los valores hacen referencia a filas existentes de *nombre-tabla-tipo*.

nombre-vista-tipo

El nombre de una vista con tipo. El tipo de datos de *nombre-columna* debe ser REF(*S*), donde *S* es el tipo de *nombre-vista-tipo* (SQLSTATE 428DM). No se realiza ninguna comprobación de los valores existentes de *nombre-columna* para asegurarse de si realmente los valores hacen referencia a filas existentes de *nombre-vista-tipo*.

SET GENERATED

Especifica si DB2 va a generar valores para la columna siempre o sólo cuando se necesite un valor por omisión.

ALWAYS

Se generará siempre un valor para la columna cuando se inserte o

se actualice una fila en la tabla. La columna debe estar ya definida como columna generada (SQLSTATE 42837).

BY DEFAULT

Se generará el valor para la columna cuando se inserte o se actualice una fila en la tabla especificando **DEFAULT** para la columna, a no ser que se especifique un valor explícito. La columna debe estar ya definida como columna generada (SQLSTATE 42837).

RESTART o RESTART WITH *constante-numérica*

Restablece el estado de la secuencia asociada con la columna de identidad. Si no se especifica **WITH** *constante-numérica*, la secuencia para la columna de identidad se reiniciará en el valor que se había especificado, implícita o explícitamente, como valor de inicio cuando se creó originalmente la columna de identidad.

La columna debe existir en la tabla especificada (SQLSTATE 42703) y ya tiene que estar definida con el atributo **IDENTITY** (SQLSTATE 42837). **RESTART** no cambia el valor original de **START WITH**.

La *constante-numérica* es una constante numérica exacta que puede ser cualquier valor positivo o negativo que se puede asignar a dicha columna (SQLSTATE 42820) a condición de que no existan dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 42894). La *constante-numérica* se utilizará como el siguiente valor para la columna.

SET INCREMENT BY *constante-numérica*

Especifica el intervalo existente entre valores consecutivos de la columna de identidad. El siguiente valor que se deberá generar para la columna de identidad se determinará a partir del último valor asignado con el incremento aplicado. La columna debe estar ya definida con el atributo **IDENTITY** (SQLSTATE 42837).

Este valor es cualquier valor positivo o negativo que pueda asignarse a esta columna (SQLSTATE 42820), que no exceda el valor de una constante entera grande (SQLSTATE 42815) y que no tenga dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA).

Si este valor es negativo, se trata de una secuencia descendente después de la sentencia **ALTER**. Si este valor es 0 o el valor absoluto es mayor que **MAXVALUE - MINVALUE** o es positivo, se trata de una secuencia ascendente después de la sentencia **ALTER**.

SET MINVALUE *constante-numérica* **o NO MINVALUE**

Especifica el valor mínimo en el que una columna de identidad

ALTER TABLE

descendente pasa por un ciclo o deja de generar valores o bien el valor en el que una columna de identidad ascendente pasa por un ciclo después de alcanzar el valor máximo. La columna debe existir en la tabla especificada (SQLSTATE 42703) y ya tiene que estar definida con el atributo IDENTITY (SQLSTATE 42837).

MINVALUE *constante-numérica*

Especifica el valor mínimo de la constante numérica. Este valor puede ser cualquier valor positivo o negativo que pueda asignarse a esta columna (SQLSTATE 42820), que no tenga dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA) pero que sea un valor inferior al valor máximo (SQLSTATE 42815).

NO MINVALUE

Para una secuencia ascendente, el valor es el valor de inicio original. Para una secuencia descendente, el valor es el valor mínimo del tipo de datos de la columna.

SET MAXVALUE *constante-numérica* **o** **NO MAXVALUE**

Especifica el valor máximo en el que una columna de identidad ascendente pasa por un ciclo o deja de generar valores o bien el valor en el que una columna de identidad descendente pasa por un ciclo después de alcanzar el valor mínimo. La columna debe existir en la tabla especificada (SQLSTATE 42703) y ya tiene que estar definida con el atributo IDENTITY (SQLSTATE 42837).

MAXVALUE *constante-numérica*

Especifica la constante numérica que es el valor máximo. Este valor puede ser cualquier valor positivo o negativo que pueda asignarse a esta columna (SQLSTATE 42820), que no tenga dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA) pero que sea un valor mayor que el valor mínimo (SQLSTATE 42815).

NO MAXVALUE

Para una secuencia ascendente, el valor es el valor máximo del tipo de datos de la columna. Para una secuencia descendente, el valor es el valor de inicio original.

SET CYCLE **o** **NO CYCLE**

Especifica si esta columna de identidad debe continuar generando valores después de generar el valor máximo o el valor mínimo. La columna debe existir en la tabla especificada (SQLSTATE 42703) y ya tiene que estar definida con el atributo IDENTITY (SQLSTATE 42837).

CYCLE

Especifica que se continúen generando valores para esta

columna después de haber alcanzado el valor máximo o el valor mínimo. Si se utiliza esta opción, cuando una columna de identidad ascendente ha alcanzado el valor máximo, genera el valor mínimo; o cuando una columna de secuencia descendente ha alcanzado el valor mínimo, genera el valor máximo. Los valores máximo y mínimo para la columna de identidad determinan el rango que se utiliza para el ciclo.

Cuando CYCLE está en vigor, se pueden generar valores duplicados para una columna de identidad. Aunque no es necesario, si se desean valores exclusivos, se puede definir un índice de unicidad de una sola columna utilizando la columna de identidad para asegurar la unicidad. Si existe un índice de unicidad en una columna de identidad de este tipo y se genera un valor no exclusivo, se producirá un error (SQLSTATE 23505).

NO CYCLE

Especifica que no se generarán valores para la columna de identidad una vez que se haya alcanzado el valor máximo o el valor mínimo.

SET CACHE *constante-entera* o NO CACHE

Especifica si se deben mantener algunos valores preasignados en memoria para obtener un acceso más rápido. Esta opción se utiliza para el rendimiento y el ajuste. La columna debe estar ya definida con el atributo IDENTITY (SQLSTATE 42837).

CACHE *constante-entera*

Especifica cuántos valores de la secuencia de identidad están preasignados y mantenidos en memoria. Cuando se generan valores para la columna de identidad, la preasignación y el almacenamiento de valores en la antememoria reducen la E/S síncrona en las anotaciones cronológicas.

Si se necesita un valor nuevo para la columna de identidad y no hay ningún valor no utilizado disponible en la antememoria, será necesario esperar a que se realice E/S en las anotaciones cronológicas para poder asignar el valor. Sin embargo, cuando se necesita un valor nuevo para la columna de identidad y existe un valor no utilizado en la antememoria, la asignación de dicho valor de identidad puede suceder más rápidamente evitando la E/S en las anotaciones cronológicas.

Cuando se detiene un gestor de bases de datos (por ejemplo, por desactivación de la base de datos, anomalía del sistema o cierre), todos los valores de secuencia en antememoria que no se han utilizado en sentencias confirmadas se pierden (es decir, no se utilizarán nunca). El valor especificado para la

ALTER TABLE

opción **CACHE** es el número máximo de valores para la columna de identidad que puede perderse en caso de anomalía del sistema.

El valor mínimo es 2 (SQLSTATE 42615).

NO CACHE

Especifica que no se debe preasignar los valores para la columna de identidad.

Si se especifica esta opción, los valores de la columna de identidad no se colocan en la memoria intermedia. En este caso, cada petición de un valor de identidad nuevo produce E/S síncrona en las anotaciones cronológicas.

SET ORDER o NO ORDER

Especifica si los valores de la columna de identidad deben generarse según el orden de petición. La columna debe existir en la tabla especificada (SQLSTATE 42703) y ya tiene que estar definida con el atributo **IDENTITY** (SQLSTATE 42837).

ORDER

Especifica que los valores de la columna de identidad se generan según el orden de petición.

NO ORDER

Especifica que los valores de la columna de identidad no necesitan generarse según el orden de petición.

DROP PRIMARY KEY

Elimina la definición de la clave primaria y todas las restricciones de referencia dependientes de esta clave primaria. La tabla debe tener una clave primaria.

DROP FOREIGN KEY *nombre-restricción*

Elimina la restricción de referencia *nombre-restricción*. El *nombre-restricción* debe identificar una restricción de referencia. Para obtener información sobre las implicaciones de eliminar una restricción de referencia, consulte el apartado "Notas" en la página 591.

DROP UNIQUE *nombre-restricción*

Elimina la definición de la restricción de unicidad *nombre-restricción* y todas las restricciones de referencia que dependen de esta restricción de unicidad. El *nombre-restricción* debe identificar una restricción **UNIQUE** existente. Para obtener información sobre las consecuencias de eliminar una restricción de unicidad, vea "Notas" en la página 591.

DROP CHECK *nombre-restricción*

Elimina la restricción de comprobación *nombre-restricción*. El *nombre-restricción* debe identificar una restricción de comprobación que esté definida en la tabla.

DROP CONSTRAINT *nombre-restricción*

Elimina la restricción *nombre-restricción*. El *nombre-restricción* debe identificar una restricción de comprobación existente, restricción de referencia, clave primaria o restricción de unicidad definida en la tabla. Para obtener información sobre las consecuencias de eliminar una restricción, vea “Notas” en la página 591.

DROP PARTITIONING KEY

Elimina la clave de particionamiento. La tabla debe tener una clave de particionamiento y debe estar en un espacio de tablas definido en un grupo de nodos de una sola partición.

DATA CAPTURE

Indica si se ha de grabar en el registro cronológico información adicional para la duplicación de datos.

Si la tabla es una tabla con tipo, entonces esta opción no está soportadas (SQLSTATE 428DH para tablas raíz o 428DR para otras subtablas).

NONE

Indica que no se va a anotar ninguna información adicional.

CHANGES

Indica que en el archivo de anotaciones se escribirá información adicional referente a los cambios de SQL efectuados en esta tabla. Esta opción es necesaria si se duplicará la tabla y se utiliza el programa Capture para capturar los cambios destinados a esta tabla y registrados en el archivo de anotaciones.

Si la tabla está definida para permitir datos en una partición que no es la partición del catálogo (grupo de nodos de múltiples particiones o grupo de nodos con una partición que no es la del catálogo), entonces, no se da soporte a esta partición (SQLSTATE 42997).

Si el nombre de esquema (implícito o explícito) de la tabla es más largo que 18 bytes, entonces no se da soporte a esta opción (SQLSTATE 42997).

Se puede encontrar más información cerca de la duplicación en el manual *Administration Guide* y en el manual *Replication Guide and Reference*.

INCLUDE LONGVAR COLUMNS

Permite que los programas de utilidad de duplicación de datos capten los cambios realizados en las columnas LONG VARCHAR o LONG VARCHARIC. La cláusula se puede especificar para las tablas que no tengan columnas LONG VARCHAR ni LONG VARCHARIC puesto que es posible MODIFICAR la tabla para que incluya estas columnas.

ALTER TABLE

ACTIVATE NOT LOGGED INITIALLY

Activa el atributo NOT LOGGED INITIALLY de la tabla para esta unidad de trabajo actual. La tabla debe haberse creado originalmente con el atributo NOT LOGGED INITIALLY (SQLSTATE 429AA).

Cualquier cambio realizado en la tabla mediante INSERT, DELETE, UPDATE, CREATE INDEX, DROP INDEX o ALTER TABLE en la misma unidad de trabajo después de que esta sentencia haya alterado la tabla no se anota cronológicamente. Se anotan cronológicamente los cambios del catálogo del sistema realizados por una sentencia ALTER en la que esté activado el atributo NOT LOGGED INITIALLY. Se anotan cronológicamente los subsiguientes cambios realizados en la misma unidad de trabajo en la información del catálogo del sistema.

Cuando se completa la unidad de trabajo actual, se desactiva el atributo NOT LOGGED INITIALLY y se anotan cronológicamente todas las operaciones realizadas en la tabla en unidades de trabajo subsiguientes.

Si se utiliza esta opción para evitar bloqueos en las tablas del catálogo mientras se insertan datos, es importante que sólo se especifique esta cláusula en la sentencia ALTER TABLE. La utilización de cualquier otra cláusula en la sentencia ALTER TABLE dará como resultado bloqueos de catálogo. Si no se especifican otras cláusulas para la sentencia ALTER TABLE, sólo se conseguirá un bloqueo SHARE en las tablas del catálogo del sistema. Esto puede reducir en gran medida la posibilidad de conflictos de simultaneidad en el intervalo de tiempo entre cuando se ejecuta esta sentencia y cuando finaliza la unidad de trabajo en la que se ha ejecutado.

Si la tabla es una tabla con tipo, esta opción sólo recibe soporte en la tabla raíz de la jerarquía de tablas con tipo (SQLSTATE 428DR).

Para obtener más información sobre el atributo NOT LOGGED INITIALLY, consulte la descripción de este atributo en "CREATE TABLE" en la página 835.

Nota: Si una tabla se ha modificado activando el atributo NOT LOGGED INITIALLY dentro de una unidad de trabajo, las peticiones de retrotraer hasta punto de salvar se convertirán en peticiones de retrotraer hasta unidad de trabajo (SQLSTATE 40506). Un error en cualquier operación de la unidad de trabajo en la que esté activo el atributo NOT LOGGED INITIALLY dará como resultado la retrotracción de toda la unidad de trabajo (SQLSTATE 40506). Además, la tabla para la que se ha activado el atributo NOT LOGGED INITIALLY queda **marcada como inaccesible después de que se haya producido la retrotracción** y sólo puede eliminarse.

Por lo tanto, debe minimizarse toda oportunidad de errores dentro de la unidad de trabajo en la que se haya activado el atributo NOT LOGGED INITIALLY.

WITH EMPTY TABLE

Causa la eliminación de todos los datos que se encuentran actualmente en una tabla. Una vez eliminados los datos, no pueden recuperarse a no ser que se utilice el recurso RESTORE. Si la unidad de trabajo en la que se ha emitido esta sentencia Alter se retrotrae, los datos de la tabla NO volverán a su estado original.

Cuando se solicite esta acción, no se activará ningún desencadenante DELETE definido en la tabla afectada. También se vaciará cualquier índice que exista en la tabla.

PCTFREE *entero*

Indica qué porcentaje de cada página se va a dejar como espacio libre durante la carga o la reorganización. El valor de *entero* entra en un rango que va de 0 a 99. La primera fila de cada página se añade sin restricciones. Cuando se añaden filas adicionales, en cada página se deja, como mínimo, un *entero* como porcentaje de espacio libre. El valor PCTFREE sólo se tiene en cuenta en los programas de utilidad LOAD y REORGANIZE TABLE. Si la tabla es una tabla con tipo, esta opción sólo recibe soporte en la tabla raíz de la jerarquía de tablas con tipo (SQLSTATE 428DR).

LOCKSIZE

Indica el tamaño (granularidad) de los bloqueos utilizados cuando se accede a la tabla. La utilización de esta opción en la definición de la tabla no evitará que se produzca un descenso escalonado de bloqueos normal. Si la tabla es una tabla con tipo, esta opción sólo recibe soporte en la tabla raíz de la jerarquía de tablas con tipo (SQLSTATE 428DR).

ROW

Indica la utilización de bloqueos de fila. Éste es el tamaño de bloqueo por omisión cuando se crea una tabla.

TABLE

Indica la utilización de bloqueos de tabla. Esto significa que se consigue el compartimiento adecuado o bloqueo exclusivo en la tabla y no se utilizan intentos de bloqueo (excepto el valor de ningún intento). La utilización de este valor puede mejorar el rendimiento de las consultas al limitarse el número de bloqueos que deben conseguirse. No obstante, también se reduce la simultaneidad porque todos los bloqueos están mantenidos sobre la tabla completa.

Puede encontrar más información sobre el bloqueo en el manual *Administration Guide*.

ALTER TABLE

APPEND

Indica si los datos se añaden al final de los datos de la tabla o si se colocan donde haya espacio libre disponible en las páginas de datos. Si la tabla es una tabla con tipo, esta opción sólo recibe soporte en la tabla raíz de la jerarquía de tablas con tipo (SQLSTATE 428DR).

ON

Indica que los datos de la tabla se añadirán y no se conservará la información sobre el espacio libre de las páginas. La tabla no debe tener un índice agrupado (SQLSTATE 428CA).

OFF

Indica que los datos de la tabla se colocarán donde haya espacio disponible. Éste es el valor por omisión cuando se crea una tabla.

Se debe reorganizar la tabla después de establecer APPEND OFF porque la información sobre el espacio libre disponible no es exacta y puede dar como resultado un rendimiento bajo durante la inserción.

VOLATILE

Esto indica al optimizador que la cardinalidad de la tabla *nombre-tabla* puede variar significativamente durante la ejecución, de vacía a bastante grande. Para acceder a *nombre-tabla*, el optimizador utilizará una exploración de índice en vez de una exploración de tabla, sean cual sean las estadísticas, si esta índice es sólo índice (todas las columnas referencias están en el índice) o este índice puede aplicar un predicador en la exploración el índice. Si la tabla es una tabla con tipo, esta opción sólo recibe soporte en la tabla raíz de la jerarquía de tablas con tipo (SQLSTATE 428DR).

NOT VOLATILE

Esto indica al optimizador que la cardinalidad de *nombre-tabla* no es volátil. Los Planes de acceso a esta tabla seguirán basándose en las estadísticas existentes y en nivel de optimización que tiene lugar.

CARDINALITY

Palabra clave opcional que indica que es el número de filas de la tabla que es volátil y no la propia tabla.

Normas

- Una columna de clave de particionamiento puede actualizarse, a menos que el parámetro de configuración DB2_UPDATE_PART_KEY se haya establecido en 'OFF' (SQLSTATE 42997). 'OFF' es el valor por omisión.
- Cualquier restricción de clave primaria o de unicidad definida en la tabla debe ser un superconjunto de la clave de particionamiento, si hay una (SQLSTATE 42997).
- Una columna que permita nulos de una clave de particionamiento puede incluirse como columna de clave foránea cuando se define la relación con

ON DELETE SET NULL, a no ser que el parámetro de configuración DB2_UPDATE_PART_KEY se haya establecido en 'OFF' (SQLSTATE 42997).

- Únicamente se puede hacer referencia a una columna en una cláusula ADD o ALTER COLUMN de una sola sentencia ALTER TABLE (SQLSTATE 42711).
- La longitud de una columna no puede alterarse si la tabla tiene tablas de resumen que dependan de la tabla (SQLSTATE 42997).
- Antes de añadir una columna generada, se debe poner la tabla en el estado de comprobación pendiente, utilizando la sentencia SET INTEGRITY (SQLSTATE 55019).

Notas

- Si se modifica una tabla para convertirla en una tabla de resumen, la tabla pasará al estado de comprobación pendiente. Si la tabla está definida como REFRESH IMMEDIATE, se debe sacar la tabla del estado de comprobación pendiente para poder invocar mandatos INSERT, DELETE o UPDATE para la tabla referenciada por la selección completa. La tabla se puede sacar del estado de comprobación pendiente utilizando REFRESH TABLE o SET INTEGRITY, con la opción IMMEDIATE CHECKED, para renovar completamente los datos de la tabla de acuerdo con la selección completa. Si los datos de la tabla reflejan fielmente el resultado de la selección completa, se puede utilizar la opción IMMEDIATE UNCHECKED de SET INTEGRITY para sacar la tabla del estado de comprobación pendiente.
- Si se modifica una tabla para convertirla en una tabla de resumen definida como REFRESH IMMEDIATE, se invalidarán todos los paquetes que estén sujetos a una operación INSERT, DELETE o UPDATE en la tabla referenciada por la selección completa.
- Si se modifica una tabla de resumen para convertirla en una tabla regular (DEFINITION ONLY), se invalidarán todos los paquetes que dependan de la tabla.
- Las cláusulas ADD se procesan antes que todas las demás cláusulas. Las demás cláusulas se procesan en el orden en que se especifican.
- Ninguna columna añadida mediante ALTER TABLE se añadirá automáticamente a ninguna vista existente de la tabla.
- Cuando se crea un índice automáticamente para una restricción de clave primaria o de unicidad, el gestor de bases de datos intentará utilizar el nombre de la restricción especificada como el nombre de índice con un nombre de esquema que coincida con el nombre de la tabla. Si esto coincide con un nombre de índice existente o no se ha especificado ningún nombre para la restricción, el índice se crea en el esquema SYSIBM con un nombre generado por el sistema y formado por "SQL" seguido de una secuencia de 15 caracteres numéricos, generados por una función basada en la indicación de la hora.

ALTER TABLE

- Se dice que cualquier tabla que participe en una operación DELETE de la tabla T está *conectada-con-supresión* a T. Así, una tabla está conectada con supresión a T si es dependiente de T o es dependiente de una tabla en la que se realizan supresiones de T en cascada.
- Un paquete tiene una utilización de inserción (actualización/supresión) en la tabla T si los registros se insertan en (actualizan en/suprimen de) T, directamente por una sentencia en el paquete, o indirectamente por las restricciones o los desencadenantes ejecutados por el paquete en nombre de una de sus sentencias. De manera similar, un paquete tiene una utilización de actualización sobre una columna si la columna se modifica directamente por una sentencia del paquete, o indirectamente por las restricciones o los desencadenantes ejecutados por el paquete en nombre de una de sus sentencias.
- Los cambios en la clave primaria, claves de unicidad o claves foráneas pueden tener el siguiente efecto en los paquetes, índices y otras claves foráneas.
 - Si se añade una clave primaria o una clave de unicidad:
 - No surte ningún efecto en los paquetes, claves foráneas o clave de unicidad existentes.⁶¹
 - Si se elimina una clave primaria o una clave de unicidad:
 - El índice se elimina si se ha creado automáticamente para la restricción. Cualquier paquete dependiente del índice se invalida.
 - El índice se vuelve a establecer como de no unicidad si se había convertido en de unicidad para la restricción y el sistema ya no lo sigue necesitando. Cualquier paquete dependiente del índice se invalida.
 - El índice se establece como ya no necesario para el sistema si era un índice de unicidad existente utilizado para la restricción. No tiene ningún efecto en los paquetes.
 - Se eliminan todas las claves foráneas dependientes. Se realiza una acción adicional para cada clave foránea dependiente, como se especifica en el punto siguiente.
 - Si se añade o se elimina una clave foránea:
 - Se invalidan todos los paquetes que estén sujetos a inserción en la tabla de objetos.
 - Se invalidan todos los paquetes que estén sujetos a actualización en una columna como mínimo de la clave foránea.
 - Se invalidan todos los paquetes que estén sujetos a supresión en la tabla padre.

61. Si la clave primaria o de unicidad utiliza un índice de unicidad existente que se ha creado en una versión anterior y no se ha convertido para dar soporte a exclusividad diferida, entonces el índice se convierte y los paquetes con utilización de actualización en la tabla asociada se invaliden.

- Se invalidan todos los paquetes que estén sujetos a actualización en una columna como mínimo de la clave padre.
- La adición de una columna a una tabla dará como resultado la invalidación de todos los paquetes sujetos a inserción en la tabla alterada. Si la columna añadida es la primera columna de tipo estructurado definido por el usuario de la tabla, también se invalidarán los paquetes sujetos a DELETE en la tabla modificada.
- La adición de una restricción de comprobación o de referencia a una tabla que ya existe y no está en estado pendiente de comprobación (vea “SET INTEGRITY” en la página 1168) hará que las filas existentes de la tabla se evalúen inmediatamente respecto a la restricción. Si la verificación resulta anómala, se emitirá un error (SQLSTATE 23512). Si una tabla está en estado pendiente de comprobación, la adición de una restricción de comprobación o de referencia no conducirá de inmediato a forzar la restricción. En cambio, se actualizarán los distintivos de tipo de restricción utilizados en la operación pendiente de comprobación. Para empezar a imponer la restricción, tendrá que emitirse la sentencia SET INTEGRITY.
- La adición o eliminación de una restricción de comprobación dará como resultado la invalidación de todos los paquetes que tengan una utilización de inserción en la tabla de objetos o una utilización de actualización en como mínimo una de las columnas implicadas en la restricción.
- La adición de una clave de particionamiento producirá la invalidación de todos los paquetes sujetos a actualización en al menos una columna de la clave de particionamiento.
- Una clave de particionamiento que esté predefinida como primera columna de la clave primaria no se ve afectada por la eliminación de la clave primaria y la adición de una clave primaria diferente.
- La alteración de una columna para aumentar la longitud invalidará todos los paquetes que hagan referencia a la tabla (directa o indirectamente a través de un desencadenante o una restricción de referencia) con la columna alterada.
- La alteración de una columna para aumentar la longitud regenerará las vistas (excepto las vistas con tipo) que dependan de la tabla. Si se produce algún error durante la regeneración de una vista, se devuelve un error (SQLSTATE 56098). Las vistas con tipo dependientes de la tabla se marcan como no operativas.
- La alteración de una columna para aumentar la longitud puede causar errores (SQLSTATE 54010) al procesar desencadenantes cuando se prepare o enlace una sentencia que involucre al desencadenante. Esto puede ocurrir cuando la longitud de fila basada en la suma de las longitudes de las variables de transición y las columnas de tabla de transición es demasiado larga. Si se elimina este desencadenante, un intento subsiguiente para crearlo producirá un error (SQLSTATE 54040).

ALTER TABLE

- Las columnas de tipo VARCHAR y VARGRAPHIC que se han modificado para ser mayores que 4000 y 2000 respectivamente, no se deben utilizar como parámetros de entrada en las funciones del esquema SYSFUN (SQLSTATE 22001).
- Cambiar LOCKSIZE para una tabla dará como resultado una invalidación de todos los paquetes que dependan de la tabla alterada. Puede encontrar más información sobre el bloqueo en el manual *Administration Guide*.
- La cláusula ACTIVATE NOT LOGGED INITIALLY no se puede utilizar cuando las columnas DATALINK con el atributo FILE LINK CONTROL se añaden a la tabla (SQLSTATE 42613).
- Al cambiar VOLATILE o NOT VOLATILE CARDINALITY se obtendrá como resultado una invalidación de todos los paquetes que tienen una dependencia en la tabla alterada.
- Los clientes de duplicación han de actuar con precaución cuando aumenten la longitud de las columnas VARCHAR. La tabla de datos de cambio asociada con una tabla de aplicaciones podría estar en el límite de tamaño de fila de DB2 o bien podría acercarse al mismo. La tabla de datos de cambio se ha de alterar antes de alterar la tabla de aplicaciones o bien se han de alterar las dos dentro de la misma unidad de trabajo, para asegurarse de que la modificación puede realizarse para ambas tablas. Hay que tener en cuenta las copias, que también pueden estar en el límite de tamaño de fila o bien cerca del mismo o bien residir en plataformas que carezcan de la característica para aumentar la longitud de una columna existente.

Si la tabla de datos de cambio no se altera antes de que el programa Capture procese los registros de anotación cronológica con la longitud de columna VARCHAR aumentada, posiblemente el programa Capture falle. Si una copia que contiene la columna VARCHAR no se modifica antes de que se ejecute la suscripción que mantiene la copia, posiblemente la suscripción fallará.

- También se soporta la sintaxis siguiente: NOMINVALUE, NOMAXVALUE, NOCYCLE, NOCACHE y NOORDER.

Ejemplos

Ejemplo 1: Añada una nueva columna llamada RATING, que tiene un carácter de longitud, a la tabla DEPARTMENT.

```
ALTER TABLE DEPARTMENT  
ADD RATING CHAR(1)
```

Ejemplo 2: Añada una nueva columna llamada SITE_NOTES a la tabla PROJECT. Cree SITE_NOTES como una columna de longitud variable con una longitud máxima de 1000 caracteres. Los valores de la columna no tienen un juego de caracteres asociado y, por lo tanto, no deben convertirse.


```
ALTER TABLE PROJECT
ADD SITE_NOTES VARCHAR(1000) FOR BIT DATA
```

Ejemplo 3: Supongamos que existe una tabla llamada EQUIPMENT definida con las siguientes columnas:

Nombre columna	Tipo datos
EQUIP_NO	INT
EQUIP_DESC	VARCHAR(50)
LOCATION	VARCHAR(50)
EQUIP_OWNER	CHAR(3)

Añada una restricción de referencia a la tabla EQUIPMENT, de manera que el propietario (EQUIP_OWNER) sea un número de departamento (DEPTNO) que esté presente en la tabla DEPARTMENT. DEPTNO es la clave primaria de la tabla DEPARTMENT. Si se elimina un departamento de la tabla DEPARTMENT, los valores del propietario (EQUIP_OWNER) referentes a todo el equipo propiedad de dicho departamento deben desasignarse (o establecerse en nulo). Dé a la restricción el nombre de DEPTQUIP.

```
ALTER TABLE EQUIPMENT
ADD CONSTRAINT DEPTQUIP
FOREIGN KEY (EQUIP_OWNER)
REFERENCES DEPARTMENT
ON DELETE SET NULL
```

Además, se necesita una columna adicional para permitir el registro de la cantidad asociada con este registro de equipo. A menos que se especifique lo contrario, la columna EQUIP_QTY debe tener un valor de 1 y nunca debe ser nulo.

```
ALTER TABLE EQUIPMENT
ADD COLUMN EQUIP_QTY
SMALLINT NOT NULL DEFAULT 1
```

Ejemplo 4: Modifique la tabla EMPLOYEE. Añada la restricción de comprobación denominada REVENUE, definida de tal manera que cada empleado debe recibir una cantidad total, entre el salario y la comisión, superior a 3.000.000 de pesetas.

```
ALTER TABLE EMPLOYEE
ADD CONSTRAINT REVENUE
CHECK (SALARY + COMM > 300000)
```

Ejemplo 5: Modifique la tabla EMPLOYEE. Elimine la restricción REVENUE que se ha definido anteriormente.

```
ALTER TABLE EMPLOYEE
DROP CONSTRAINT REVENUE
```

Ejemplo 6: Modifique una tabla para anotar cronológicamente los cambios SQL en el formato por omisión.

ALTER TABLE

```
ALTER TABLE SALARY1
DATA CAPTURE NONE
```

Ejemplo 7: Modifique una tabla para anotar cronológicamente los cambios SQL en un formato expandido.

```
ALTER TABLE SALARY2
DATA CAPTURE CHANGES
```

Ejemplo 8: Modifique la tabla EMPLOYEE para añadir 4 nuevas columnas con los valores por omisión.

```
ALTER TABLE EMPLOYEE
ADD COLUMN HEIGHT MEASURE DEFAULT MEASURE(1)
ADD COLUMN BIRTHDAY BIRTHDATE DEFAULT DATE('01-01-1850')
ADD COLUMN FLAGS BLOB(1M) DEFAULT BLOB(X'01')
ADD COLUMN PHOTO PICTURE DEFAULT BLOB(X'00')
```

Los valores por omisión utilizan varios nombres de función al especificar el valor por omisión. Debido a que MEASURE es un tipo diferenciado basado en INTEGER, se utiliza la función MEASURE. El valor por omisión de la columna HEIGHT se podría haber especificado sin la función debido a que el tipo fuente de MEASURE no es BLOB ni un tipo de datos de fecha y hora. Debido a que BIRTHDATE es un tipo diferenciado basado en DATE, se utiliza la función DATE (BIRTHDATE no puede utilizarse aquí). Para las columnas FLAGS y PHOTO el valor por omisión se especifica utilizando la función BLOB aunque PHOTO es un tipo diferenciado. Para especificar un valor por omisión para las columnas BIRTHDAY, FLAGS y PHOTO, se debe utilizar una función porque el tipo es BLOB o bien un tipo diferenciado derivado de un tipo de datos BLOB o de fecha y hora.

Ejemplo 9: Suponga que tiene una tabla llamada CUSTOMERS que está definida con las siguientes columnas:

Nombre columna	Tipo datos
BRANCH_NO	SMALLINT
CUSTOMER_NO	DECIMAL(7)
CUSTOMER_NAME	VARCHAR(50)

En esta tabla, la clave primaria está formada por las columnas BRANCH_NO y CUSTOMER_NO. Se desea particionar la tabla, por lo tanto necesita crear una clave de particionamiento para la tabla. La tabla debe definirse en un espacio de tablas de un grupo de nodos de un solo nodo. La clave primaria debe ser un superconjunto de las columnas de partición: como mínimo una de las columnas de la clave primaria debe utilizarse como la clave de particionamiento. Suponga que desea que BRANCH_NO sea la clave de particionamiento. Podría hacerlo con la sentencia siguiente:

```
ALTER TABLE CUSTOMERS
ADD PARTITIONING KEY (BRANCH_NO)
```

ALTER TABLESPACE

La sentencia ALTER TABLESPACE se utiliza para modificar un espacio de tablas existente de las siguientes maneras.

- Añadir un contenedor a un espacio de tablas DMS (es decir, un espacio de tablas creado con la opción MANAGED BY DATABASE).
- Aumentar el tamaño de un contenedor del espacio de tablas DMS (es decir, un espacio de tablas creado con la opción MANAGED BY DATABASE).
- Añadir un contenedor a un espacio de tablas SMS de una partición (o nodo) que no tenga contenedores actualmente.
- Modificar el valor PREFETCHSIZE para un espacio de tablas.
- Modificar el valor BUFFERPOOL utilizado para las tablas del espacio de tablas.
- Modificar el valor OVERHEAD para un espacio de tablas.
- Modificar el valor TRANSFERRATE para un espacio de tablas.

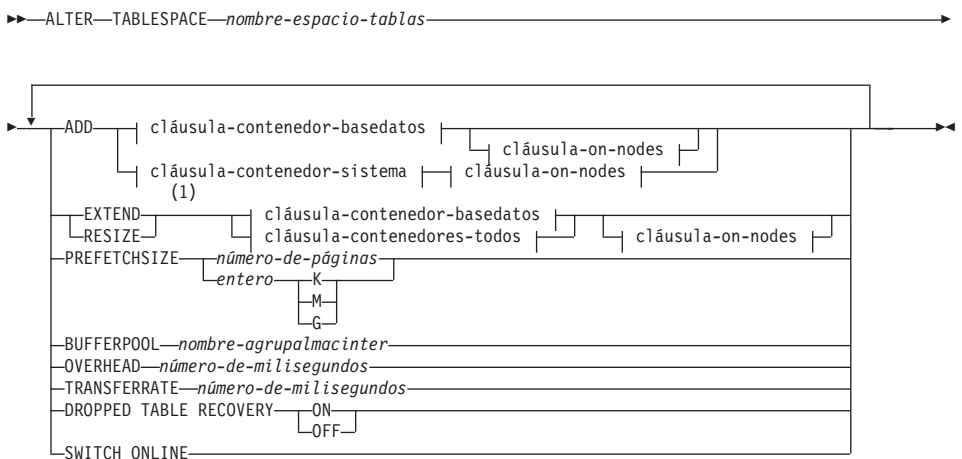
Invocación

Esta sentencia puede intercalarse en un programa de aplicación o emitirse de forma interactiva. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar dinámicamente (SQLSTATE 42509).

Autorización

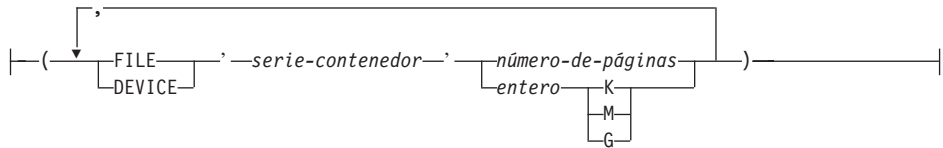
El ID de autorización de la sentencia debe tener autorización SYSCTRL o SYSADM.

Sintaxis



ALTER TABLESPACE

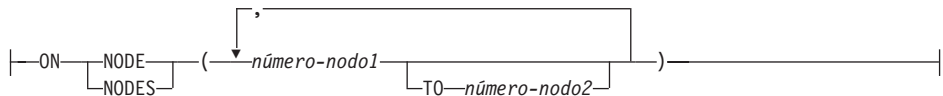
cláusula-contenedor-basedatos:



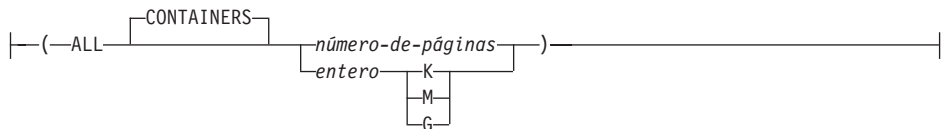
cláusula-contenedor-sistema:



cláusula-on-nodes:



cláusula-contenedores-todos:



Notas:

- 1 Las cláusulas ADD, EXTEND y RESIZE no se pueden especificar en una misma sentencia.

Descripción

nombre-espacio-tablas

Designa el espacio de tablas. Es un nombre que consta de una sola parte. Es un identificador SQL largo (ordinario o delimitado).

ADD

ADD especifica que va a añadir un nuevo contenedor al espacio de tablas.

EXTEND

EXTEND especifica que se aumenta el tamaño de los contenedores existentes. El tamaño especificado es el tamaño en el que se aumenta el

contenedor existente. Si se especifica la *cláusula-contenedores-todos*, se aumenta el tamaño de todos los contenedores del espacio de tablas.

RESIZE

RESIZE especifica la modificación del tamaño de los contenedores existentes (el tamaño de los contenedores sólo se puede aumentar). El tamaño especificado es el nuevo tamaño del contenedor. Si se especifica la *cláusula-contenedores-todos*, se cambian a ese tamaño todos los contenedores del espacio de tablas.

cláusula-contenedor-basedatos

Añade uno o varios contenedores a un espacio de tablas DMS. El espacio de tablas debe identificar un espacio de tablas DMS que ya exista en el servidor de aplicaciones. Consulte la descripción de *cláusula-contenedor* en la página 897.

cláusula-contenedor-sistema

Añade uno o más contenedores a un espacio de tablas SMS en las particiones o nodos especificados. El espacio de tablas debe identificar un espacio de tablas SMS que ya exista en el servidor de aplicaciones. No debe haber ningún contenedor en las particiones especificadas para el espacio de tablas (SQLSTATE 42921). Consulte la descripción de *contenedores-sistema* en la página 896.

cláusula-on-nodes

Especifica la partición o particiones para los contenedores añadidos. Consulte la descripción de *cláusula-on-nodes* en la página 898.

cláusula-contenedores-todos

Amplía o cambia el tamaño de todos los contenedores del espacio de tablas DMS. El espacio de tablas debe identificar un espacio de tablas DMS que ya exista en el servidor de aplicaciones.

PREFETCHSIZE *número-de-páginas*

Especifica el número de páginas de PAGESIZE que se leerán del espacio de tablas cuando se realice la lectura anticipada de datos. El valor del tamaño de lectura anticipada también puede especificarse como un valor entero seguido de K (para kilobytes), M (para megabytes) o G (para gigabytes). Si se especifica de esta manera, el nivel mínimo del número de bytes dividido por el tamaño de página se utiliza para determinar el valor del número de páginas para el tamaño de lectura anticipada. La lectura anticipada lee los datos que una consulta necesita antes que la consulta haga referencia a ellos, por lo que la consulta no necesita esperar a que se efectúen operaciones de E/S.

BUFFERPOOL *nombre-agrupalmacinter*

El nombre de la agrupación de almacenamientos intermedios utilizado para las tablas de este espacio de tablas. La agrupación de almacenamientos intermedios debe existir actualmente en la base de datos

ALTER TABLESPACE

(SQLSTATE 42704). El grupo de nodos del espacio de tablas debe estar definido para la agrupación de almacenamientos intermedios (SQLSTATE 42735).

OVERHEAD *número-de-milisegundos*

Cualquier literal numérico (entero, decimal o de coma flotante) que especifique la actividad general del controlador de E/S y el tiempo de búsqueda y latencia del disco, en milisegundos. Para todos los contenedores que pertenecen al espacio de tablas, este número debe ser un promedio (o, en todo caso, el mismo número). Este valor sirve para determinar el coste de E/S durante la optimización de una consulta.

TRANSFERRATE *número-de-milisegundos*

Cualquier literal numérico (entero, decimal o de coma flotante) que especifique el tiempo para leer una página (de 4K u 8K) en la memoria en milisegundos. Para todos los contenedores que pertenecen al espacio de tablas, este número debe ser un promedio (o, en todo caso, el mismo número). Este valor sirve para determinar el coste de E/S durante la optimización de una consulta.

DROPPED TABLE RECOVERY

Las tablas eliminadas del espacio de tablas especificado pueden recuperarse mediante la opción RECOVER DROPPED TABLE ON del mandato ROLLFORWARD.

SWITCH ONLINE

Los espacios de tablas en estado OFFLINE se ponen en línea (ONLINE) si los contenedores han pasado a ser accesibles. Si los contenedores no son accesibles, se devuelve un error (SQLSTATE 57048).

Notas

- Se proporciona ayuda para elegir los valores óptimos para los parámetros PREFETCHSIZE, OVERHEAD y TRANSFERRATE e información sobre el equilibrio en el manual *Administration Guide*.
- Después de añadir el nuevo contenedor y confirmar la transacción, el contenido del espacio de tablas se redistribuye automáticamente entre los contenedores. El acceso al espacio de tablas no está restringido durante la redistribución del contenido.
- Si el espacio de tablas está en estado OFFLINE y los contenedores se convierten en accesibles, el usuario puede desconectar todas las aplicaciones y conectarse de nuevo a la base de datos para que el espacio de tablas salga del estado OFFLINE. Alternativamente, la opción SWITCH ONLINE puede activar el espacio de tablas (fuera del estado OFFLINE) mientras que el resto de la base de datos sigue activo y se utiliza.
- Si se añade más de un contenedor a un espacio de tablas, es recomendable añadirlos en la misma sentencia, para que la redistribución del espacio sólo deba hacerse una vez. Si se intenta añadir contenedores a un mismo espacio

de tablas en sentencias ALTER TABLESPACE diferentes dentro de una sola transacción, se producirá un error (SQLSTATE 55041).

- No se puede cambiar el tamaño de un contenedor de un espacio de tablas y añadir nuevos contenedores mediante una sola sentencia ALTER TABLESPACE (SQLSTATE 429BC). Si se cambia el tamaño de más de un contenedor, no se pueden utilizar simultáneamente las cláusulas EXTEND y RESIZE en una sola sentencia (SQLSTATE 429BC).
- RESIZE no se puede utilizar para disminuir el tamaño de contenedores. Si se intenta especificar un tamaño menor para un contenedor, se producirá un error (SQLSTATE 560B0).
- Si se intenta ampliar o cambiar el tamaño de contenedores que no existen, se producirá un error (SQLSTATE 428B2).
- Cuando se amplía o cambia el tamaño de un contenedor, el tipo de contenedor debe coincidir con el tipo que se utilizó al crear el contenedor (SQLSTATE 428B2).
- Después de ampliar o cambiar el tamaño de un contenedor, y una vez confirmada la transacción, el contenido del espacio de tablas se redistribuye automáticamente entre todos los contenedores. El acceso al espacio de tablas no está restringido durante la redistribución del contenido.
- Si se amplían varios contenedores de un espacio de tablas, es recomendable cambiarlos en la misma sentencia, para que la redistribución del espacio sólo deba hacerse una vez. Esto también es aplicable al cambio de tamaño de varios contenedores. Si se intenta cambiar el tamaño de contenedores de un mismo espacio de tablas, utilizando sentencias ALTER TABLESPACE diferentes dentro de una sola transacción, se producirá un error (SQLSTATE 55041).
- En una base de datos particionada si más de una partición reside en el mismo nodo físico, no se puede especificar el mismo dispositivo o vía de acceso específica para dichas particiones (SQLSTATE 42730). Para este entorno, especifique una *serie-contenedor* exclusiva para cada partición o utilice una vía de acceso relativa.
- Aunque la definición del espacio de tablas es transaccional y los cambios de la definición del espacio de tablas se reflejarán en las tablas de catálogo al realizar la confirmación, no puede utilizarse la agrupación de almacenamientos intermedios con la nueva definición hasta la próxima vez que se inicie la base de datos. La agrupación de almacenamientos intermedios en uso, cuando se ha emitido la sentencia ALTER TABLESPACE, continuará utilizándose mientras tanto.

Ejemplos

Ejemplo 1: Añada un dispositivo al espacio de tablas PAYROLL.

```
ALTER TABLESPACE PAYROLL
  ADD (DEVICE '/dev/rhdisk9' 10000)
```

ALTER TABLESPACE

Ejemplo 2: Cambie el tamaño de lectura anticipada y la actividad general de E/S para el espacio de tablas ACCOUNTING.

```
ALTER TABLESPACE ACCOUNTING
  PREFETCHSIZE 64
  OVERHEAD 19.3
```

Ejemplo 3: Cree un espacio de tablas TS1, luego cambie el tamaño de todos los contenedores a 2000 páginas (se proporcionan tres sentencias ALTER TABLESPACES diferentes para realizar este cambio de tamaño).

```
CREATE TABLESPACE TS1
  MANAGED BY DATABASE
  USING (FILE '/conts/cont0' 1000,
        DEVICE '/dev/rcont1' 500,
        FILE 'cont2' 700)
ALTER TABLESPACE TS1
  RESIZE (FILE '/conts/cont0' 2000,
        DEVICE '/dev/rcont1' 2000,
        FILE 'cont2' 2000)
```

○

```
ALTER TABLESPACE TS1
  RESIZE (ALL 2000)
```

○

```
ALTER TABLESPACE TS1
  EXTEND (FILE '/conts/cont0' 1000,
        DEVICE '/dev/rcont1' 1500,
        FILE 'cont2' 1300)
```

Ejemplo 4: Amplíe en 1000 páginas todos los contenedores del espacio de tablas DATA_TS.

```
ALTER TABLESPACE DATA_TS
  EXTEND (ALL 1000)
```

Ejemplo 5: Cambie a 100 megabytes (MB) el tamaño de todos los contenedores del espacio de tablas INDEX_TS .

```
ALTER TABLESPACE INDEX_TS
  RESIZE (ALL 100 M)
```


ALTER TYPE (Estructurado)

La sentencia ALTER TYPE se utiliza para añadir o eliminar especificaciones de atributo o de método de un tipo estructurado definido por el usuario.

Invocación

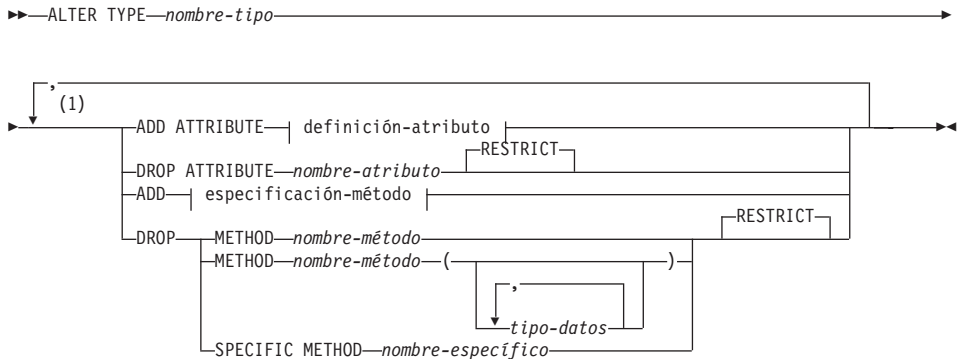
Esta sentencia se puede intercalar en un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámicas. Es una sentencia ejecutable que se puede preparar dinámicamente. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no puede prepararse dinámicamente (SQLSTATE 42509).

Autorización

El ID de autorización de la sentencia debe tener como mínimo uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Privilegio ALTERIN para el esquema del tipo.
- Definidor del tipo, tal como está registrado en la columna DEFINER de SYSCAT.DATATYPES

Sintaxis



Notas:

- 1 Si se añaden o eliminan tanto atributos como métodos, las especificaciones de atributo deben preceder a las especificaciones de método.

Descripción

nombre-tipo

Identifica el tipo estructurado a cambiar. Debe ser un tipo existente definido en el catálogo (SQLSTATE 42704) y el tipo debe ser un tipo estructurado (SQLSTATE 428DP). En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un

ALTER TYPE (Estructurado)

nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objetos no calificados.

ADD ATTRIBUTE

Añade un atributo después del último atributo del tipo estructurado existente.

definición-atributo

Para obtener una descripción detallada de *definición-atributo*, vea "CREATE TYPE (Estructurado)" en la página 923.

nombre-atributo

Especifica un nombre para el atributo. El nombre no puede ser el mismo que el de otro atributo de este tipo estructurado (incluidos los atributos heredados) ni de ningún subtipo de este tipo estructurado (SQLSTATE 42711).

Algunos nombres utilizados como palabras clave en predicados están reservados para su uso por el sistema, y no pueden utilizarse como *nombre-atributo* (SQLSTATE 42939). Estos nombres son SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH y los operadores de comparación.

tipo-datos 1

Especifica el tipo de datos del atributo. Es uno de los tipos de datos listados en la descripción de CREATE TABLE, excepto LONG VARCHAR, LONG VARGRAPHIC, o un tipo diferenciado basado en LONG VARCHAR o LONG VARGRAPHIC (SQLSTATE 42601). El tipo de datos debe identificar un tipo de datos existente (SQLSTATE 42704). Si *tipo-datos* está especificado sin un nombre de esquema, el tipo se resuelve realizando una búsqueda en los esquemas de la vía de acceso de SQL. La descripción de diversos tipos de datos se proporciona en el apartado "CREATE TABLE" en la página 835. Si el tipo de datos del atributo es un tipo de referencia, el tipo destino de la referencia debe ser un tipo estructurado existente (SQLSTATE 42704).

Si un tipo estructurado está definido con un atributo de tipo DATALINK, sólo puede utilizarse de manera efectiva como tipo de datos para una tabla con tipo o vista con tipo (SQLSTATE 01641).

Para evitar que las definiciones de tipo que, durante la ejecución, permitirían que una instancia del tipo contenga, directa o indirectamente, otra instancia del mismo tipo o uno de sus subtipos, existe una restricción que impide definir un tipo de forma que uno de sus tipos de atributo haga uso de sí mismo

directa o indirectamente (SQLSTATE 428EP). Vea “Tipos estructurados” en la página 97 para obtener más información.

opciones-lob

Especifica las opciones correspondientes a tipos LOB (o tipos diferenciados basados en tipos LOB). Para obtener una descripción detallada de opciones-lob, vea “CREATE TABLE” en la página 835.

opciones-datalink

Especifica las opciones asociadas con tipos DATALINK (o tipos diferenciados basados en tipos DATALINK). Para obtener una descripción detallada de opciones-datalink, vea “CREATE TABLE” en la página 835.

Observe que, si no se especifican opciones para un tipo DATALINK o un tipo diferenciado derivado de DATALINK, las opciones LINKTYPE URL y NO LINK CONTROL serán las opciones por omisión.

DROP ATTRIBUTE

Elimina un atributo del tipo estructurado existente.

nombre-atributo

El nombre del atributo. El atributo debe existir como un atributo del tipo (SQLSTATE 42703).

RESTRICT

Hace cumplir la regla de que no se puede eliminar ningún atributo si *nombre-tipo* es utilizado como tipo de una tabla, vista, columna, atributo anidado dentro del tipo de una columna o de una extensión de índice.

ADD especificación-método

Añade una especificación de método al tipo identificado por nombre-tipo. Para poder utilizar el método es necesario darle un cuerpo mediante una sentencia CREATE METHOD separada. Para obtener más información sobre la especificación de método, vea “CREATE TYPE (Estructurado)” en la página 923.

DROP METHOD

Identifica una instancia de un método que se debe eliminar. El método especificado no debe tener un cuerpo de método existente (SQLSTATE 428ER). Utilice la sentencia DROP METHOD para eliminar el cuerpo del método antes de utilizar ALTER TYPE DROP METHOD.

El método especificado debe estar descrito en el catálogo (SQLSTATE 42704). Los métodos generados implícitamente por la sentencia CREATE TYPE (tales como mutadores y observadores) no se pueden eliminar (SQLSTATE 42917).

ALTER TYPE (Estructurado)

Hay varias maneras para identificar la especificación de método que se debe eliminar:

METHOD *nombre-método*

Identifica un método determinado, y sólo es válido si hay exactamente una sola instancia de método cuyo nombre sea *nombre-método* y su tipo sea *nombre-tipo*. El método así identificado puede tener un número cualquiera de parámetros. Si no existe ningún método con ese nombre para el tipo *nombre-tipo*, se produce un error (SQLSTATE 42704). Si existe más de un método con el nombre *nombre-método* para el tipo de datos indicado, se produce un error (SQLSTATE 42854).

METHOD *nombre-método (tipo-datos,...)*

Proporciona la signatura del método, que identifica de manera exclusiva el método que se ha de eliminar. El algoritmo de selección de métodos no se utiliza.

nombre-método

Es el nombre del método que se debe eliminar correspondiente al tipo específico. El nombre debe ser un identificador no calificado.

(tipo-datos, ...)

Deben coincidir con los tipos de datos que se especificaron en las posiciones correspondientes de la especificación de método cuando se definió el método. El número de tipos de datos y la concatenación lógica de los tipos de datos se utiliza para identificar la instancia de método específica que se debe eliminar.

Para los tipos de datos parametrizados no es necesario especificar la longitud, la precisión ni la escala. En su lugar, se puede codificar un conjunto vacío de paréntesis para indicar que estos atributos se deben pasar por alto al buscar un tipo de datos determinado.

No puede utilizarse `FLOAT()` (SQLSTATE 42601) puesto que el valor del parámetro indica tipos de datos diferentes (REAL o DOUBLE).

Sin embargo, si se codifica la longitud, la precisión o la escala, el valor debe coincidir exactamente con el especificado en la sentencia `CREATE TYPE`.

El tipo de datos `FLOAT(n)` no necesita coincidir con el valor definido para `n`, pues $0 < n < 25$ significa REAL y $24 < n < 54$ significa DOUBLE. La coincidencia se produce basándose en si el tipo es REAL o DOUBLE.

Si no existe ningún método que tenga la signatura especificada para el tipo de datos indicado, se produce un error (SQLSTATE 42883).

SPECIFIC METHOD *nombre-específico*

Identifica el método determinado que se debe eliminar, utilizando el nombre específico que se proporcionó o que se tomó por omisión cuando se definió el método. Si *nombre-específico* es un nombre no calificado, el método se califica implícitamente mediante el esquema del tipo de datos especificado para *nombre-tipo*. El nombre-específico debe identificar un método correspondiente al tipo *nombre-tipo*; de lo contrario se produce un error (SQLSTATE 42704).

RESTRICT

Indica que el método especificado no puede tener un cuerpo de método existente. Utilice la sentencia DROP METHOD para eliminar el cuerpo del método antes de utilizar ALTER TYPE DROP METHOD.

Normas

- La adición o eliminación de un atributo no está permitida para el tipo *nombre-tipo* (SQLSTATE 55043) si se cumple cualquiera de estas condiciones:
 - el tipo o uno de sus subtipos es el tipo de una tabla o vista existente
 - existe una columna de una tabla cuyo tipo utiliza, directa o indirectamente, *nombre-tipo*. Los términos *utiliza directamente* y *utiliza indirectamente* están definidos en “Tipos estructurados” en la página 97
 - el tipo o uno de sus subtipos se utiliza en extensión de índice.
- No se puede modificar un tipo mediante la adición de atributos si el número total de atributos para el tipo o cualquiera de sus subtipos es mayor que 4082 (SQLSTATE 54050).
- Opción ADD ATTRIBUTE:
 - ADD ATTRIBUTE genera métodos de observador y mutador para el nuevo atributo. Estos métodos son similares a los generados cuando se crea un tipo estructurado, tal como se describe en “CREATE TYPE (Estructurado)” en la página 923. Si estos métodos invalidan o entran en conflicto con métodos o funciones existentes, la sentencia ALTER TYPE falla (SQLSTATE 42745).
 - Si el usuario especificó explícitamente un valor menor que 292 para INLINE LENGTH del tipo (o de cualquiera de sus subtipos), y los atributos añadidos hacen que esa longitud especificada sea menor que el tamaño del resultado de la función constructora para el tipo modificado (32 bytes más 10 bytes por cada atributo), se produce un error (SQLSTATE 42611).
- Opción DROP ATTRIBUTE:
 - Un atributo que se ha heredado de un supertipo existente no se puede eliminar (SQLSTATE 428DJ).
 - DROP ATTRIBUTE elimina los métodos de mutador y observador de los atributos eliminados y comprueba si hay dependencias respecto a esos métodos eliminados.

ALTER TYPE (Estructurado)

Notas

- Cuando se altera un tipo, mediante la adición o eliminación de un atributo, se invalidan todos los paquetes que dependen de funciones o métodos que utilizan este tipo o un subtipo de él como parámetro o resultado.
- Cuando un atributo se añade a un tipo estructurado o se elimina de él:
 - Si el valor `INLINE LENGTH` del tipo fue calculado por el sistema cuando se creó el tipo, los valores `INLINE LENGTH` se modifican automáticamente para el tipo alterado, y para todos sus subtipos, para reflejar el cambio. Los valores `INLINE LENGTH` también se modifican automáticamente (recursivamente) para todos los tipos estructurados cuando `INLINE LENGTH` fue calculado por el sistema y el tipo incluye un atributo de cualquier tipo con un valor `INLINE LENGTH` cambiado.
 - Si el usuario especificó explícitamente el valor `INLINE LENGTH` de un tipo cualquiera que se alteró mediante la adición o eliminación de atributos, entonces el valor `INLINE LENGTH` de ese tipo determinado no se modifica. Debe tenerse especial cuidado en el caso de valores `INLINE LENGTH` especificados explícitamente. Si es probable que más tarde se añadan atributos a un tipo, el valor de `INLINE LENGTH`, para cualquier uso de ese tipo o de uno de sus subtipos en una definición de columna, debe ser lo suficiente grande para tener en cuenta el posible aumento de longitud del objeto del que se creó una instancia.
 - Si deben habilitarse nuevos atributos para ser utilizados por programas de aplicación, se deben modificar las funciones de transformación existentes para que coincidan con la nueva estructura del tipo de datos.

Ejemplos

Ejemplo 1: La sentencia `ALTER TYPE` puede utilizarse para permitir un ciclo de tablas y tipos que se hacen referencia mutuamente. Piense en las tablas denominadas `EMPLOYEE` y `DEPARTMENT` que se hacen referencia mutuamente.

La secuencia siguiente permitiría crear los tipos y las tablas.

```
CREATE TYPE DEPT ...
CREATE TYPE EMP ... (incluido el atributo denominado DEPTREF del tipo REF(DEPT))
ALTER TYPE DEPT ADD ATTRIBUTE MANAGER REF(EMP)
CREATE TABLE DEPARTMENT OF DEPT ...
CREATE TABLE EMPLOYEE OF EMP (DEPTREF WITH OPTIONS SCOPE DEPARTMENT)
ALTER TABLE DEPARTMENT ALTER COLUMN MANAGER ADD SCOPE EMPLOYEE
```

La secuencia siguiente permitiría eliminar estas tablas y tipos.

```
DROP TABLE EMPLOYEE (la columna MANAGER de DEPARTMENT se queda sin ámbito)
DROP TABLE DEPARTMENT
ALTER TYPE DEPT DROP ATTRIBUTE MANAGER
DROP TYPE EMP
DROP TYPE DEPT
```

Ejemplo 2: La sentencia ALTER TYPE puede utilizarse para crear un tipo con un atributo que haga referencia a un subtipo.

```
CREATE TYPE EMP ...
CREATE TYPE MGR UNDER EMP ...
ALTER TYPE EMP ADD ATTRIBUTE MANAGER REF(MGR)
```

Ejemplo 3: La sentencia ALTER TYPE puede utilizarse para añadir un atributo. La sentencia siguiente añade el atributo SPECIAL al tipo EMP. Debido a que la sentencia CREATE TYPE original no especificaba el valor INLINE LENGTH, DB2 lo recalcula añadiendo 13 bytes (10 bytes para el nuevo atributo + longitud del atributo + 2 bytes para un atributo que no es LOB).

```
ALTER TYPE EMP ...
ADD ATTRIBUTE SPECIAL CHAR(1)
```

Ejemplo 4: La sentencia ALTER TYPE puede utilizarse para añadir un método asociado a un tipo. La sentencia siguiente añade un método llamado BONUS.

```
ALTER TYPE EMP ...
ADD METHOD BONUS (RATE DOUBLE)
RETURNS INTEGER
LANGUAGE SQL
CONTAINS SQL
NO EXTERNAL ACTION
DETERMINISTIC
```

Observe que para utilizar el método BONUS es necesario emitir una sentencia CREATE METHOD para crear el cuerpo del método. Si se supone que el tipo EMP incluye un atributo llamado SALARY, el ejemplo siguiente define un cuerpo de método.

```
CREATE METHOD BONUS(RATE DOUBLE) FOR EMP
RETURN CAST(SELF.SALARY * RATE AS INTEGER)
```

Vea “CREATE METHOD” en la página 790 para obtener una descripción de esta sentencia.

ALTER USER MAPPING

ALTER USER MAPPING

La sentencia ALTER USER MAPPING se utiliza para cambiar el ID de autorización o la contraseña que se utilizan en una fuente de datos para el ID de autorización de un servidor federado especificado.

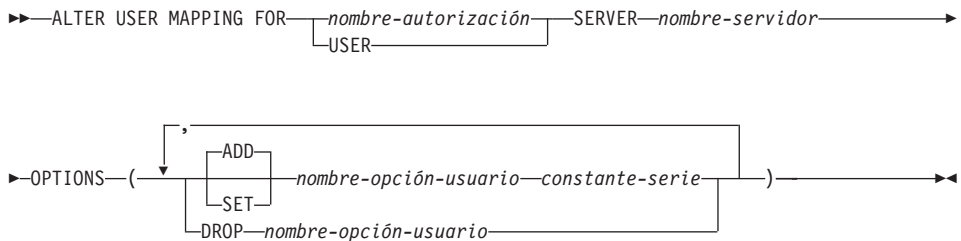
Invocación

Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

Autorización

Si el ID de autorización de la sentencia es diferente del nombre de autorización que se correlaciona con la fuente de datos, el ID de autorización de la sentencia debe incluir la autorización SYSADM o DBADM. De lo contrario, si el ID de autorización y el nombre de autorización coinciden, no es necesario ningún privilegio ni ninguna autorización.

Sintaxis



Descripción

nombre-autorización

Especifica el nombre de autorización bajo el cual un usuario o una aplicación se conecta a una base de datos federada.

USER

El valor del registro especial USER. Cuando se especifica USER, el ID de autorización de la sentencia ALTER USER MAPPING se correlacionará con el ID de autorización de la fuente de datos que se especifica en la opción de usuario REMOTE_AUTHID.

SERVER *nombre-servidor*

Identifica la fuente de datos que se puede acceder bajo el ID de autorización remoto que se correlaciona con el ID de autorización local que indica *nombre-autorización* o al que USER hace referencia.

OPTIONS

Indica las opciones de usuario que se han de habilitar, restaurar o desactivar para la correlación que se está modificando. Consulte el apartado “Opciones de usuario” en la página 1420 para ver las descripciones de *nombre-opción-usuario* y sus valores.

ADD

Habilita una opción de usuario.

SET

Cambia el valor de una opción de usuario.

nombre-opción-usuario

Nombra una opción de usuario que se ha de habilitar o restaurar.

constante-serie

Especifica el valor para *nombre-opción-usuario* como una constante de serie de caracteres.

DROP *nombre-opción-usuario*

Desactiva una opción de usuario.

Notas

- Una opción de usuario no se puede especificar más de una vez en la misma sentencia ALTER USER MAPPING (SQLSTATE 42853). Cuando se habilita, restaura o desactiva una opción de usuario, no afecta a ninguna otra opción de usuario que se esté utilizando.
- Una correlación de usuarios no se puede modificar en una unidad de trabajo (UOW) determinada si la UOW ya incluye una sentencia SELECT que hace referencia a un apodo para una tabla o una vista de la fuente de datos que se ha de incluir en la correlación.

Ejemplos

Ejemplo 1: Jim utiliza una base de datos local para conectarse a una fuente de datos Oracle llamada ORACLE1. Accede a la fuente de datos local bajo el ID de autorización KLEEWEIN; KLEEWEIN se correlaciona con CORONA, el ID de autorización bajo el cual accede a ORACLE1. Jim va a iniciar el acceso a ORACLE1 bajo un nuevo ID, JIMK. De modo que ahora es necesario correlacionar KLEEWEIN con JIMK.

```
ALTER USER MAPPING FOR KLEEWEIN
SERVER ORACLE1
OPTIONS ( SET REMOTE_AUTHID 'JIMK' )
```

Ejemplo 2: Mary utiliza una base de datos federada para conectarse a una fuente de datos DB2 Universal Database para OS/390 llamada DORADO. Utiliza un ID de autorización para acceder a DB2 y otro para acceder a DORADO y ha creado una correlación entre los dos ID. Ha utilizado la misma contraseña con ambos ID, pero ahora decide utilizar una contraseña diferente,

ALTER USER MAPPING

ZNYQ, con el ID para DORADO. De acuerdo a ello, necesita correlacionar la contraseña de base de datos federada con ZNYQ.

```
ALTER USER MAPPING FOR MARY
  SERVER DORADO
  OPTIONS ( ADD REMOTE_PASSWORD 'ZNYQ' )
```

ALTER VIEW

La sentencia ALTER VIEW modifica una vista existente alterando una columna de tipo de referencia para añadir un ámbito.

Invocación

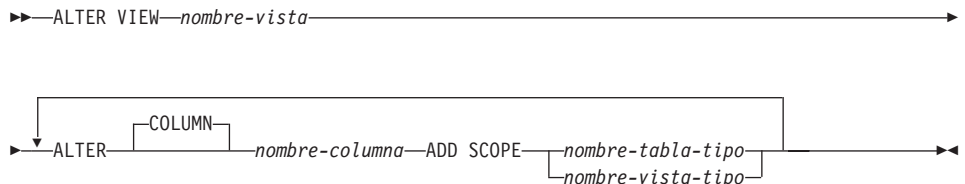
Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que se puede preparar dinámicamente. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar dinámicamente (SQLSTATE 42509).

Autorización

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Privilegio ALTERIN para el esquema de la vista
- Definidor de la vista que se debe alterar
- Privilegio CONTROL para la vista que se debe alterar.

Sintaxis



Descripción

nombre-vista

Identifica la vista que se va a cambiar. Debe ser una vista descrita en el catálogo.

ALTER COLUMN *nombre-columna*

Es el nombre de la columna que se va a alterar en la vista. El *nombre-columna* debe identificar una columna existente de la vista (SQLSTATE 42703). El nombre no puede estar calificado.

ADD SCOPE

Añade un ámbito a una columna de tipo de referencia existente que todavía no tiene un ámbito definido (SQLSTATE 428DK). La columna no debe ser herencia de una supervista (SQLSTATE 428DJ).

nombre-tabla-tipo

El nombre de una tabla con tipo. El tipo de datos de *nombre-columna*

ALTER VIEW

debe ser REF(*S*), donde *S* es el tipo de *nombre-tabla-tipo* (SQLSTATE 428DM). No se realiza ninguna comprobación de los valores existentes de *nombre-columna* para asegurarse de si realmente los valores hacen referencia a filas existentes de *nombre-tabla-tipo*.

nombre-vista-tipo

El nombre de una vista con tipo. El tipo de datos de *nombre-columna* debe ser REF(*S*), donde *S* es el tipo de *nombre-vista-tipo* (SQLSTATE 428DM). No se realiza ninguna comprobación de los valores existentes de *nombre-columna* para asegurarse de si realmente los valores hacen referencia a filas existentes de *nombre-vista-tipo*.

BEGIN DECLARE SECTION

La sentencia BEGIN DECLARE SECTION marca el principio de una sección de declaración de variables del lenguaje principal.

Invocación

Esta sentencia sólo puede incluirse en un programa de aplicación. No es una sentencia ejecutable. No debe especificarse en REXX.

Autorización

No se necesita.

Sintaxis

►►—BEGIN DECLARE SECTION—◄◄

Descripción

La sentencia BEGIN DECLARE SECTION puede codificarse en el programa de aplicación siempre que puedan aparecer declaraciones de variables de acuerdo a las reglas del lenguaje principal. Se utiliza para indicar el inicio de una sección de declaración de variables del lenguaje principal. Una sección de variables del lenguaje principal finaliza con una sentencia END DECLARE SECTION (vea “END DECLARE SECTION” en la página 1034).

Normas

- Las sentencias BEGIN DECLARE SECTION y END DECLARE SECTION deben especificarse por pares y no pueden anidarse.
- No pueden incluirse sentencias de SQL dentro de la sección de declaración.
- Las variables a las que sentencias de SQL hagan referencia deben declararse en una sección de declaración en todos los lenguajes principales que no sean REXX. Además, la sección debe aparecer antes que la primera referencia a la variable. Por lo general, las variables del lenguaje principal no se declaran en REXX, excepto los localizadores de LOB y las variables de referencia a archivos. En este caso, no se declaran dentro de una BEGIN DECLARE SECTION.
- Las variables que se declaren fuera de una sección de declaración no deberán tener el mismo nombre que las variables que se declaran en la sección de declaración.
- El tipo de datos y la longitud de los tipos de datos LOB deben ir precedidos por las palabras clave SQL TYPE IS.

Ejemplos

Ejemplo 1: Defina las variables de lenguaje principal hv_smint (smallint), hv_vchar24 (varchar(24)), hv_double (double), hv_blob_50k (blob(51200)), hv_struct (del tipo estructurado "struct_type" como blob(10240)) en un programa escrito en C.

BEGIN DECLARE SECTION

```
EXEC SQL BEGIN DECLARE SECTION;      short          hv_smint;
      struct {
          short hv_vchar24_len;
          char  hv_vchar24_value[24];
      }
      hv_vchar24;
      double          hv_double;
      SQL TYPE IS BLOB(50K) hv_blob_50k;
      SQL TYPE IS struct_type AS BLOB(10k) hv_struct;
EXEC SQL END DECLARE SECTION;
```

Ejemplo 2: Defina las variables del lenguaje principal HV-SMINT (smallint), HV-VCHAR24 (varchar(24)), HV-DEC72 (dec(7,2)) y HV-BLOB-50k (blob(51200)) en un programa COBOL.

```
      WORKING-STORAGE SECTION.
EXEC SQL BEGIN DECLARE SECTION END-EXEC.
01  HV-SMINT          PIC S9(4)          COMP-4.
01  HV-VCHAR24.
    49 HV-VCHAR24-LENGTH PIC S9(4)          COMP-4.
    49 HV-VCHAR24-VALUE  PIC X(24).
01  HV-DEC72          PIC S9(5)V9(2)     COMP-3.
01  HV-BLOB-50K       USAGE SQL TYPE IS BLOB(50K).
EXEC SQL END DECLARE SECTION END-EXEC.
```

Ejemplo 3: Defina las variables del lenguaje principal HVSMINT (smallint), HVVCHAR24 (char(24)), HVDOUBLE (double) y HVBLOB50k (blob(51200)) en un programa Fortran.

```
EXEC SQL BEGIN DECLARE SECTION
      INTEGER*2      HVSMINT
      CHARACTER*24   HVVCHAR24
      REAL*8         HVDOUBLE
      SQL TYPE IS BLOB(50K) HVBLOB50K
EXEC SQL END DECLARE SECTION
```

Nota: En Fortran, si el valor esperado es mayor que 254 caracteres, debe utilizarse una variable del lenguaje principal CLOB.

Ejemplo 4: Defina las variables del lenguaje principal HVSMINT (smallint), HVBLOB50K (blob(51200)) y HVCLOBLOC (un localizador de CLOB) en un programa REXX.

```
      DECLARE :HVCLOBLOC LANGUAGE TYPE CLOB LOCATOR
call sqlexec 'FETCH c1 INTO :HVSMINT, :HVBLOB50K'
```

Observe que las variables HVSMINT y HVBLOB50K se han definido de manera implícita al utilizarlas en la sentencia FETCH.

CALL

Invoca un procedimiento almacenado en la ubicación de una base de datos. Un procedimiento almacenado, por ejemplo, se ejecuta en la ubicación de la base de datos y devuelve los datos a la aplicación cliente.

Los programas que utilizan la sentencia de SQL CALL se diseñan para ejecutarse en dos partes, una en el cliente y la otra en el servidor. El procedimiento del servidor en la base de datos se ejecuta en la misma transacción que la aplicación cliente. Si la aplicación cliente y el procedimiento almacenado están en la misma partición, el procedimiento almacenado se ejecuta localmente.

Invocación

Esta sentencia sólo puede incluirse en un programa de aplicación. Se trata de una sentencia ejecutable que no puede prepararse de forma dinámica. No obstante, el nombre de procedimiento puede especificarse mediante una variable del lenguaje principal y esto, junto con el uso de la cláusula USING DESCRIPTOR, permite proporcionar tanto el nombre de procedimiento como la lista de parámetros en tiempo de ejecución; con lo que se consigue el mismo efecto que con una sentencia preparada dinámicamente.

Autorización

Las reglas de autorización varían de acuerdo con el servidor en el que está almacenado el procedimiento.

DB2 Universal Database:

El ID de autorización de la sentencia CALL debe tener como mínimo uno de los privilegios siguientes **durante la ejecución**:

- Privilegio EXECUTE para el paquete asociado al procedimiento almacenado
- Privilegio CONTROL para el paquete asociado al procedimiento almacenado
- Autorización SYSADM o DBADM

DB2 Universal Database para OS/390:

El ID de autorización de la sentencia CALL debe tener como mínimo los privilegios siguientes **durante el enlace**:

- Privilegio EXECUTE para el paquete asociado al procedimiento almacenado
- Calidad de propietario del paquete asociado al procedimiento almacenado
- Autorización PACKADM para la colección del paquete
- Autorización SYSADM

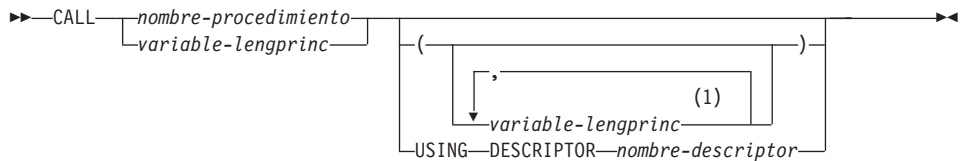
CALL

DB2 para AS/400:

El ID de autorización de la sentencia CALL debe tener como mínimo los privilegios siguientes **durante el enlace**:

- Si el procedimiento almacenado está escrito en REXX:
 - Las autorizaciones del sistema *OBJOPR y *READ para el archivo fuente asociado al procedimiento
 - La autorización del sistema *EXECUTE para la biblioteca donde reside el archivo fuente y la autorización del sistema *USE para el mandato CL
- Si el procedimiento almacenado no está escrito en REXX:
 - La autorización del sistema *EXECUTE para el programa asociado al procedimiento y para la biblioteca donde reside ese programa
- Autorización de administración

Sintaxis



Notas:

- 1 Los procedimientos almacenados ubicados en los servidores DB2 Universal Database para OS/390 y DB2 Universal Database para AS/400 e invocados por clientes DB2 Universal Database para OS/390 o DB2 Universal Database para AS/400 dan soporte a fuentes adicionales de argumentos de procedimiento (por ejemplo, valores de constante). Sin embargo, si el procedimiento almacenado está ubicado en una DB2 Universal Database o el procedimiento se invoca desde una DB2 Universal Database cliente, se deben proporcionar todos los argumentos mediante las variables del lenguaje principal.

Descripción

nombre-procedimiento o *variable-lengprinc*

Identifica el procedimiento que se va a llamar. El nombre de procedimiento puede especificarse directamente o dentro de una variable del lenguaje principal. El procedimiento identificado debe existir en el servidor actual (SQLSTATE 42724).

Si se especifica *nombre-procedimiento*, debe ser un identificador normal con un máximo de 254 bytes. Por lo tanto sólo puede ser un identificador normal, no puede contener blancos ni caracteres especiales y el valor se

convierte a mayúsculas. Así, si es necesario utilizar nombres en minúsculas, blancos o caracteres especiales, el nombre debe especificarse mediante una *variable-lengprinc*.

Si se especifica *variable-lengprinc*, debe ser una variable serie-caracteres con un atributo de longitud que tenga un máximo de 254 bytes y no debe incluir una variable indicadora. Observe que el valor **no** se convierte a mayúsculas. El *nombre-procedimiento* debe estar justificado por la izquierda.

El nombre de procedimiento puede tomar un formato entre varios. Los formatos soportados varían, según el servidor en el que está almacenado el procedimiento.

DB2 Universal Database:

nombre-procedimiento

El nombre (sin extensión) del procedimiento que se va a ejecutar. El procedimiento invocado se determina de la manera siguiente.

1. El *nombre-procedimiento* se utiliza como nombre de la biblioteca de procedimientos almacenados y el nombre de función dentro de dicha biblioteca. Por ejemplo, si el *nombre-procedimiento* es `proclib`, el servidor DB2 cargará la biblioteca de procedimientos almacenados `proclib` y ejecutará la rutina de función `proclib()` dentro de esa biblioteca.

En los sistemas basados en UNIX, el servidor DB2 encuentra la biblioteca de procedimientos almacenados en el directorio por omisión `sqllib/function`. Los procedimientos almacenados no restringidos están en el directorio `sqllib/function/unfenced`.

En OS/2, la ubicación de los procedimientos almacenados está especificada en la variable `LIBPATH` del archivo `CONFIG.SYS`. Los procedimientos almacenados no restringidos están en el directorio `sqllib\dll\unfenced`.

2. Si no se ha podido encontrar la biblioteca o función, se utiliza el *nombre-procedimiento* para buscar en los procedimientos definidos (en `SYSCAT.PROCEDURES`) uno que coincida. Un procedimiento que coincida se determina utilizando los pasos siguientes.
 - a. Busque los procedimientos del catálogo (`SYSCAT.PROCEDURES`) donde `PROCNAME` coincida con el

nombre-procedimiento especificado y PROCSCHEMA sea un nombre de esquema en la vía de acceso de SQL (registro especial CURRENT PATH). Si el nombre de esquema está especificado explícitamente, la vía de acceso de SQL se ignora y sólo se tienen en cuenta los procedimientos con el nombre de esquema especificado.

- b. Después, elimine cualquiera de estos procedimientos que no tengan el mismo número de parámetros que el número de argumentos especificados en la sentencia CALL.
- c. Elija el procedimiento restante que esté antes en la vía de acceso de SQL.
- d. Si después de finalizar el paso 2 no queda ningún procedimiento, se devuelve un error (SQLSTATE 42884).

Cuando se ha seleccionado el procedimiento, DB2 invocará el procedimiento definido por el nombre externo.

biblioteca-procedimiento!nombre-función

El carácter de admiración (!), actúa como delimitador entre el nombre de biblioteca y el nombre de función del procedimiento almacenado. Por ejemplo, si se especificase proclib!func, se cargaría proclib en memoria y se ejecutaría la función func de esa biblioteca. Esto permite que se coloquen múltiples funciones en la misma biblioteca de procedimientos almacenados.

La biblioteca de procedimientos almacenados está ubicada en los directorios o especificada en la variable LIBPATH, tal como se describe en *nombre-procedimiento*.

vía-absoluta!nombre-función

La *vía-absoluta* especifica la vía de acceso completa a la biblioteca de procedimientos almacenados.

En un sistema basado en UNIX, por ejemplo, si se especificase /u/terry/proclib!func, se obtendría la biblioteca de procedimientos almacenados proclib del directorio /u/terry y se ejecutaría la función func de esa biblioteca.

En OS/2, si se ha especificado `d:\terry\proclib!func`, haría que el gestor de bases de datos cargase el archivo `func.dll` del directorio `d:\terry\proclib`.

En todos estos casos, la longitud total del nombre de procedimiento, incluida su vía de acceso completa implícita o explícita, no debe tener una longitud superior a 254 bytes.

Servidor DB2 Universal Database para OS/390 (V4.1 ó posterior):

Un nombre implícito o explícito compuesto de tres partes. Estas partes son las siguientes:

orden superior:

El nombre de ubicación del servidor donde está almacenado el procedimiento.

medio: SYSPROC

medio: Algún valor en la columna PROCEDURE de la tabla del catálogo SYSIBM.SYSPROCEDURES.

Servidor DB2 para OS/400 (V3.1 ó posterior):

El nombre de programa externo se supone que es igual al *nombre-procedimiento*.

Por razones de portabilidad, *nombre-procedimiento* debe especificarse como un símbolo individual de 8 bytes como máximo.

(*variable-lengprinc*,...)

Cada especificación de *variable-lengprinc* es un parámetro de CALL. El parámetro *n*ésimo de CALL corresponde al parámetro *n*ésimo del procedimiento almacenado del servidor.

Se supone que se utiliza cada *variable-lengprinc* para intercambiar datos en ambas direcciones entre cliente y servidor. Para evitar enviar datos innecesarios entre cliente y servidor, la aplicación cliente debe proporcionar una variable indicadora con cada parámetro y establecer el indicador en -1 si el parámetro no se utiliza para transmitir datos al procedimiento almacenado. El procedimiento almacenado debe establecer la variable indicadora en -128 para cualquier parámetro que no se utilice para devolver datos a la aplicación cliente.

Si el servidor es DB2 Universal Database los parámetros deben tener tipos de datos coincidentes en el programa cliente y servidor.⁶²

62. Los servidores DB2 Universal Database para OS/390 y DB2 Universal Database para AS/400 dan soporte a las conversiones entre tipos de datos compatibles cuando invocan sus procedimientos almacenados. Por ejemplo, si el programa cliente utiliza el tipo de datos INTEGER y el procedimiento almacenado espera FLOAT, el servidor convertirá el valor INTEGER a FLOAT antes de invocar el procedimiento.

CALL

USING DESCRIPTOR *nombre-descriptor*

Identifica una SQLDA que debe contener una descripción válida de las variables del lenguaje principal. El enésimo elemento de la SQLVAR corresponde al enésimo parámetro del procedimiento almacenado del procedimiento almacenado del servidor.

Antes de que se procese la sentencia CALL, la aplicación debe definir los campos siguientes de la SQLDA:

- SQLN para indicar el número de apariciones de SQLVAR proporcionadas en la SQLDA
- SQLDABC para indicar el número de bytes de almacenamiento asignados para la SQLDA
- SQLD para indicar el número de variables utilizadas en la SQLDA al procesar la sentencia
- Las apariciones de SQLVAR, para indicar los atributos de las variables. Deben inicializarse los siguientes campos de cada elemento pasado de SQLVAR base:
 - SQLTYPE
 - SQLLEN
 - SQLDATA
 - SQLIND

Deben inicializarse los siguientes campos de cada elemento pasado de la SQLVAR secundaria:

- LEN.SQLLONGLEN
- SQLDATALEN
- SQLDATATYPE_NAME

Se supone que cada SQLDA se utiliza para intercambiar datos en ambas direcciones entre cliente y servidor. Para evitar enviar datos innecesarios entre cliente y servidor, la aplicación cliente debe establecer el campo SQLIND en -1 si el parámetro no se utiliza para transmitir datos al procedimiento almacenado. El procedimiento almacenado debe establecer el campo SQLIND en -128 para cualquier parámetro que no se utilice para devolver datos a la aplicación cliente.

Notas

- **Utilización de tipos de datos Gran objeto (LOB):**

Si la aplicación cliente y servidora tiene que especificar datos LOB de una SQLDA, asigne el doble al número de entradas SQLVAR.

A partir de DB2 Versión 2, los procedimientos almacenados dan soporte a los tipos de datos LOB. Estos tipos de datos no están soportados por todos los clientes o servidores de versiones anteriores.

- **Recuperación del estado de retorno (RETURN_STATUS) resultante de un procedimiento SQL:**

Si un procedimiento SQL emite satisfactoriamente una sentencia RETURN junto con un valor de estado, este valor se coloca en el primer campo SQLERRD de la SQLCA. Si la sentencia CALL se emite en un procedimiento SQL, utilice la sentencia GET DIAGNOSTICS para recuperar el valor de estado de retorno (RETURN_STATUS). El valor es -1 cuando SQLSTATE indica un error.

- **Devolución de los resultados de procedimientos almacenados:**

Si el programa de aplicación cliente se escribe utilizando CLI, los conjuntos de resultados pueden devolverse directamente a la aplicación cliente. El procedimiento almacenado indica que un conjunto resultante debe devolverse declarando un cursor en ese conjunto resultante, abriendo un cursor en el conjunto resultante y dejando el cursor abierto al salir del procedimiento.

Al final de un procedimiento que se invoca mediante CLI:

- Por cada cursor que se ha dejado abierto, se devuelve un conjunto resultante a la aplicación.
- Si se deja abierto más de un cursor, los conjuntos del resultado se devuelven en el orden en que se han abierto sus cursores.
- Sólo se devuelven las filas no leídas. Por ejemplo, si el conjunto resultante de un cursor tiene 500 filas y el procedimiento almacenado ha leído 150 de estas filas en el momento en que ha terminado el procedimiento almacenado, se devolverán las filas desde la 151 a la 500 al procedimiento almacenado.

Para obtener información adicional, consulte los manuales *Application Development Guide* y *CLI Guide and Reference*.

- **Interoperabilidad entre la sentencia CALL y la API DARI:**

En general, la sentencia CALL no funcionará con procedimientos DARI existentes. Vea el manual *Application Development Guide* para obtener detalles.

- **Manejo de registros especiales:**

Los valores de los registros especiales utilizados por el llamador son heredados por el procedimiento almacenado durante la invocación y se restauran al devolver el control al llamador. Los registros especiales se pueden modificar dentro de un procedimiento almacenado, pero estos cambios no se aplican al llamador. Esto no es cierto para los procedimientos almacenados preexistentes (aquéllos definidos con el estilo de parámetro DB2DARI o situados en la biblioteca por omisión), en los que los cambios hechos en los registros especiales de un procedimiento se convierten en los valores del llamador.

CALL

Ejemplos

Ejemplo 1:

En C, invoque un procedimiento denominado TEAMWINS en la biblioteca ACHIEVE pasándole un parámetro almacenado en la variable del lenguaje principal HV_ARGUMENT.

```
strcpy(HV_PROCNAME, "ACHIEVE!TEAMWINS");  
CALL :HV_PROCNAME (:HV_ARGUMENT);
```

Ejemplo 2:

En C, invoque un procedimiento denominado :SALARY_PROC utilizando la SQLDA denominada INOUT_SQLDA.

```
struct sqlda *INOUT_SQLDA;  
  
/* El código de configuración para variables SQLDA va aquí */  
  
CALL :SALARY_PROC  
USING DESCRIPTOR :*INOUT_SQLDA;
```

Ejemplo 3:

Un procedimiento almacenado Java se define en la base de datos utilizando la sentencia siguiente:

```
CREATE PROCEDURE PARTS_ON_HAND (IN PARTNUM INTEGER,  
                                OUT COST DECIMAL(7,2),  
                                OUT QUANTITY INTEGER)  
EXTERNAL NAME 'pieza!disponibles'  
LANGUAGE JAVA PARAMETER STYLE DB2GENERAL;
```

Una aplicación Java llama a este procedimiento almacenado utilizando el siguiente fragmento de código:

```
...  
CallableStatement stpCall ;  
  
String sql = "CALL PARTS_ON_HAND ( ?,?,? )" ;  
  
stpCall = con.prepareStatement( sql ) ; /* con es la conexión */  
  
stpCall.setInt( 1, variable1 ) ;  
stpCall.setBigDecimal( 2, variable2 ) ;  
stpCall.setInt( 3, variable3 ) ;  
  
stpCall.registerOutParameter( 2, Types.DECIMAL, 2 ) ;  
stpCall.registerOutParameter( 3, Types.INTEGER ) ;  
  
stpCall.execute() ;
```

```
variable2 = stpCall.getBigDecimal(2) ;  
variable3 = stpCall.getInt(3) ;  
...
```

Este fragmento de código de aplicación invocará el método Java *onhand* de la clase *parts* ya que el nombre-procedimiento especificado en la sentencia CALL se encuentra en la base de datos y tiene el nombre externo 'parts!onhand'.

CLOSE

CLOSE

La sentencia CLOSE cierra un cursor. Si se ha creado una tabla resultante cuando se ha abierto el cursor, se ha destruido esa tabla.

Invocación

Esta sentencia puede incluirse en un programa de aplicación o emitirse de forma interactiva. Se trata de una sentencia ejecutable que no puede prepararse de forma dinámica.

Autorización

No se necesita. Consulte “DECLARE CURSOR” en la página 976 para conocer la autorización necesaria para utilizar un cursor.

Sintaxis

```
►—CLOSE—nombre-cursor—┐  
└─WITH RELEASE─┘
```

Descripción

nombre-cursor

Identifica el cursor que se va a cerrar. El *nombre-cursor* debe identificar un cursor declarado tal como se explica en la sentencia DECLARE CURSOR. Cuando se ejecuta la sentencia CLOSE, el cursor debe estar en el estado abierto.

WITH RELEASE

Se intenta liberar todos los bloqueos leídos que se han mantenido para el cursor. Observe que no todos los bloqueos leídos se liberan necesariamente; estos bloqueos se pueden mantener para otras operaciones o actividades.

Notas

- Al final de una unidad de trabajo, todos los cursores que pertenecen a un proceso de aplicación y que se han declarado sin la opción WITH HOLD se cierran de manera implícita.
- La cláusula WITH RELEASE no tiene ningún efecto para los cursores que están funcionando bajo los niveles de aislamiento CS o UR. Cuando se especifica para cursores que están funcionando bajo los niveles de aislamiento RS o RR, WITH RELEASE termina algunas de las garantías de dichos niveles de aislamiento. De forma específica, si se vuelve a abrir el cursor, un cursor RS puede experimentar el fenómeno de ‘lectura no repetible’, y un cursor RR puede experimentar el fenómeno de ‘lectura no repetible’ y el de ‘fantasma’. Consulte el “Apéndice I. Comparación de niveles de aislamiento” en la página 1453 para obtener más detalles.

Si un cursor que originalmente era RR o RS se vuelve a abrir después de haberse cerrado mediante la cláusula `WITH RELEASE`, se adquirirán nuevos bloqueos de lectura.

- Se aplican reglas especiales a los cursores dentro de un procedimiento almacenado que no se haya cerrado antes de volver al programa que lo llama. Vea “Notas” en la página 622 para obtener más información.

Ejemplo

Se utiliza un cursor para buscar una fila cada vez en las variables del programa `C dnum, dname y mnum`. Finalmente, se cierra el cursor. Si se vuelve a abrir el cursor, se sitúa de nuevo al principio de las filas en las que se ha de buscar.

```
EXEC SQL DECLARE C1 CURSOR FOR
      SELECT DEPTNO, DEPTNAME, MGRNO
      FROM TDEPT
      WHERE ADMRDEPT = 'A00';

EXEC SQL OPEN C1;

while (SQLCODE==0) {
    EXEC SQL FETCH C1 INTO :dnum, :dname, :mnum;
    .
    .
}

EXEC SQL CLOSE C1;
```

COMMENT

COMMENT

La sentencia COMMENT añade o sustituye comentarios en las descripciones del catálogo de diversos objetos.

Invocación

Esta sentencia se puede intercalar en un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámicas. Es una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

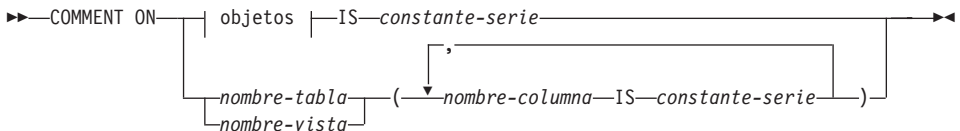
Autorización

Los privilegios que debe tener el ID de autorización de la sentencia COMMENT deben incluir uno de los siguientes:

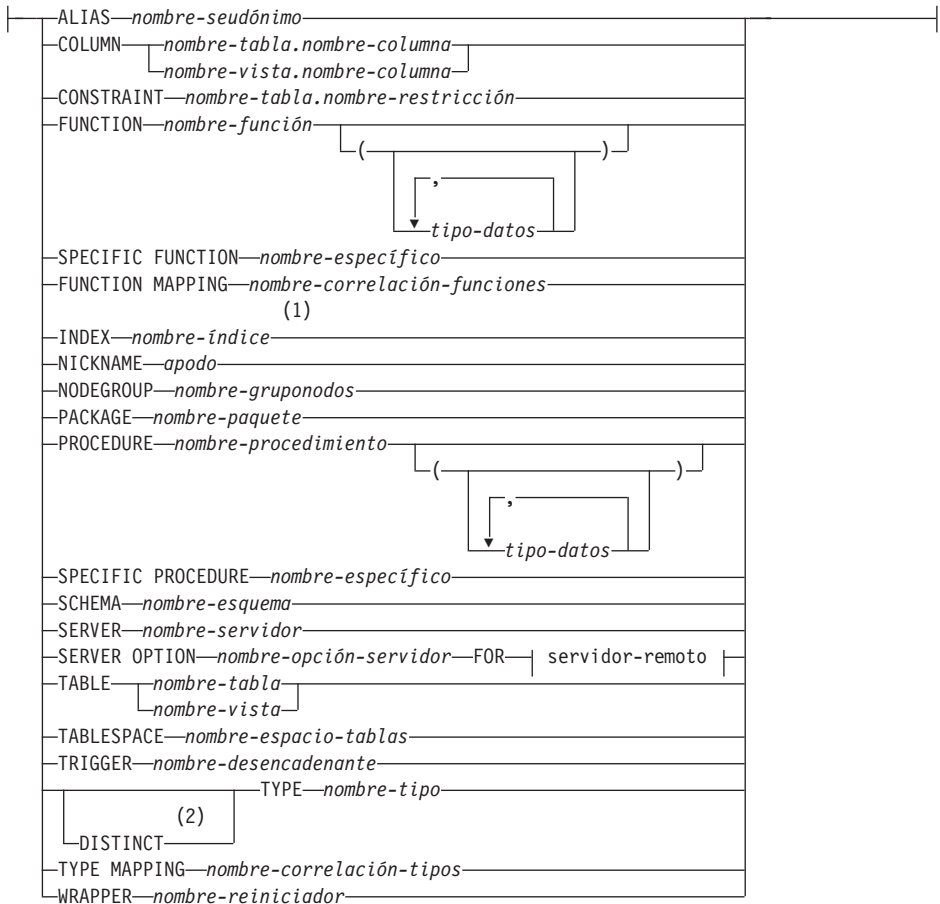
- SYSADM o DBADM
- definidor del objeto (tabla principal para la columna o restricción) tal como está registrado en la columna DEFINER de la vista de catálogo para el objeto (columna OWNER para un esquema)
- Privilegio ALTERIN para el esquema (sólo aplicable a los objetos que permiten nombres de más de una parte)
- Privilegio CONTROL para el objeto (sólo aplicable a los objetos de índice, paquete, tabla y vista)
- Privilegio ALTER para el objeto (sólo aplicable a los objetos de tabla)

Observe que para un espacio de tablas o grupo de nodos, el ID de autorización debe tener la autorización SYSADM o SYSCTRL.

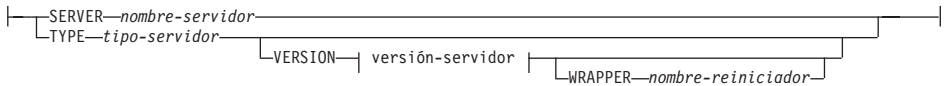
Sintaxis



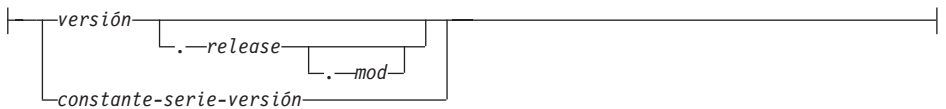
objetos:



servidor-remoto:



versión-servidor:



COMMENT

Notas:

- 1 *Nombre-índice* puede ser el nombre de un índice o una especificación de índice.
- 2 La palabra clave DATA se puede utilizar como sinónimo de DISTINCT.

Descripción

ALIAS *nombre-seudónimo*

Indica un comentario que se añadirá o sustituirá para un seudónimo. El *nombre-seudónimo* debe designar un seudónimo descrito en el catálogo (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.TABLES para la fila que describe el seudónimo.

COLUMN *nombre-tabla.nombre-columna* o *nombre-vista.nombre-columna*

Indica un comentario que se añadirá o se sustituirá para una columna. La combinación *nombre-tabla.nombre-columna nombre-vista.nombre-columna* debe identificar una combinación de columna y tabla que esté descrita en el catálogo (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.COLUMNS para la fila que describe la columna.

No puede realizarse un comentario sobre una columna de una vista no operativa. (SQLSTATE 51024).

CONSTRAINT *nombre-tabla.nombre-restricción*

Indica que se añadirá o se sustituirá un comentario para una restricción. La combinación *nombre-tabla.nombre-restricción* debe identificar una restricción y la tabla que restringe; deben estar descritos en el catálogo (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.TABCONST para la fila que describe la restricción.

FUNCTION

Indica que se añadirá o se sustituirá un comentario para una función. La instancia de función especificada debe ser una función definida por el usuario o una plantilla de función descritas en el catálogo.

Hay varias maneras distintas disponibles para identificar la instancia de función:

FUNCTION *nombre-función*

Identifica la función en particular y sólo es válida si hay exactamente una función con ese *nombre-función*. La función así identificada puede tener cualquier número de parámetros definidos para la misma. En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace QUALIFIER específica de forma implícita el calificador para los

nombres de objeto no calificados. Si no existe ninguna función con este nombre en el esquema nombrado o implícito, se produce un error (SQLSTATE 42704). Si existe más de una instancia específica de la función en el esquema mencionado o implicado, se producirá un error (SQLSTATE 42725).

FUNCTION *nombre-función (tipo-datos,...)*

Proporciona la signatura de la función, que identifica de manera exclusiva la función que hay que comentar. El algoritmo de selección de función *no* se utiliza.

nombre-función

Proporciona el nombre de la función que hay que comentar. En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objetos no calificados.

(tipo-datos,...)

Debe coincidir con los tipos de datos que se han especificado en la sentencia CREATE FUNCTION en la posición correspondiente. El número de tipos de datos y la concatenación lógica de los tipos de datos se utilizan para identificar la función específica para la que se añade o se sustituye el comentario.

Si *tipo-datos* no está calificado, el nombre de tipo se resuelve efectuando una búsqueda en los esquemas de la vía de acceso de SQL. Esto también se aplica a los nombres de tipo de datos especificados para un tipo REFERENCE.

Para los tipos de datos parametrizados no es necesario especificar la longitud, la precisión ni la escala. En su lugar, puede haber un conjunto vacío de paréntesis codificados para indicar que esos atributos deben ignorarse al buscar una coincidencia del tipo de datos.

No puede utilizarse FLOAT() (SQLSTATE 42601) puesto que el valor del parámetro indica tipos de datos diferentes (REAL o DOUBLE).

Sin embargo, si la longitud, la precisión o la escala están codificadas, el valor debe coincidir exactamente con lo especificado en la sentencia CREATE FUNCTION.

No es necesario que un tipo de FLOAT(n) coincida con el valor definido para n, pues $0 < n < 25$ significa REAL y $24 < n < 54$ significa DOUBLE. La coincidencia se produce basándose en si el tipo es REAL o DOUBLE.

COMMENT

(Observe que el atributo FOR BIT DATA no se considera parte de la signatura por lo que respecta a la selección por comparación. Así, por ejemplo, un CHAR FOR BIT DATA especificado en la signatura coincidiría con una función definida sólo con CHAR, y viceversa.)

Si en el esquema nombrado o implícito no hay ninguna función con la signatura especificada, se genera un error (SQLSTATE 42883).

SPECIFIC FUNCTION *nombre-específico*

Indica que se añadirán o se sustituirán comentarios para una función (vea FUNCTION para conocer otros métodos de identificar una función). Identifica la función en particular definida por el usuario que hay que comentar, utilizando el nombre específico que se ha especificado o tomado por omisión en el tiempo de creación de la función. En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace QUALIFIER especifica de forma implícita el calificador para los nombres de objeto no calificados. El *nombre-específico* debe identificar una instancia de función específica en el esquema nombrado o implícito; de lo contrario, se produce un error (SQLSTATE 42704).

No es posible realizar un comentario sobre una función que esté en el esquema SYSIBM o en el esquema SYSFUN (SQLSTATE 42832).

El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.FUNCTIONS para la fila que describe la función.

FUNCTION MAPPING *nombre-correlación-funciones*

Indica que se añadirá o sustituirá un comentario para la correlación de una función. El *nombre-correlación-funciones* debe identificar una correlación de funciones que está descrita en el catálogo (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.FUNCMAPPINGS correspondiente a la fila que describe la correlación de funciones.

INDEX *nombre-índice*

Indica que se añadirá o sustituirá un comentario para un índice o especificación de índice. El *nombre-índice* debe designar un índice diferenciado o una especificación de índice que esté descrita en el catálogo (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.INDEXES correspondiente a la fila que describe el índice o especificación de índice.

NICKNAME *apodo*

Indica que se añadirá o sustituirá un comentario para un apodo. El *apodo*

debe ser un apodo descrito en el catálogo (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.TABLES correspondiente a la fila que describe el apodo.

NODEGROUP *nombre-gruponodos*

Indica que se añadirá o sustituirá un comentario para un grupo de nodos. El *nombre-gruponodos* debe identificar un grupo de nodos diferenciado que esté descrito en el catálogo (SQLSTATE 42704). El comentario sustituye el valor para la columna REMARKS de la vista de catálogo SYSCAT.NODEGROUPS para la fila que describe el grupo de nodos.

PACKAGE *nombre-paquete*

Indica que se añadirá o se sustituirá un comentario para un paquete. El *nombre-paquete* debe identificar un paquete diferenciado que esté descrito en el catálogo (SQLSTATE 42704). El comentario sustituye el valor para la columna REMARKS de la vista de catálogo SYSCAT.PACKAGES para la fila que describe el paquete.

PROCEDURE

Indica que se añadirá o sustituirá un comentario para un procedimiento. La instancia del procedimiento especificado debe ser un procedimiento almacenado descrito en el catálogo.

Hay varias maneras disponibles de identificar la instancia de procedimiento:

PROCEDURE *nombre-procedimiento*

Identifica el procedimiento en particular y sólo es válido si hay exactamente un procedimiento con el *nombre-procedimiento* en el esquema. El procedimiento identificado de esta manera puede tener cualquier número de parámetros definido. En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace QUALIFIER especifica de forma implícita el calificador para los nombres de objeto no calificados. Si no existe ningún procedimiento con este nombre en el esquema nombrado o implícito, se genera un error (SQLSTATE 42704). Si existe más de una instancia específica del procedimiento en el esquema mencionado o implicado, se producirá un error (SQLSTATE 42725).

PROCEDURE *nombre-procedimiento (tipo-datos,...)*

Se utiliza para proporcionar la signatura del método, que identifica de manera exclusiva al procedimiento que se comenta.

nombre-procedimiento

Proporciona el nombre de procedimiento del procedimiento que hay que comentar. En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un

nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace QUALIFIER especifica de forma implícita el calificador para los nombres de objeto no calificados.

(tipo-datos,...)

Deben coincidir con los tipos de datos que se han especificado en la sentencia CREATE PROCEDURE en la posición correspondiente. El número de tipos de datos y la concatenación lógica de los tipos de datos se utilizan para identificar el procedimiento específico para el que se añade o se sustituye el comentario.

Si *tipo-datos* no está calificado, el nombre de tipo se resuelve efectuando una búsqueda en los esquemas de la vía de acceso de SQL. Esto también se aplica a los nombres de tipo de datos especificados para un tipo REFERENCE.

Para los tipos de datos parametrizados no es necesario especificar la longitud, la precisión ni la escala. En su lugar, puede haber un conjunto vacío de paréntesis codificados para indicar que esos atributos deben ignorarse al buscar una coincidencia del tipo de datos.

No puede utilizarse FLOAT() (SQLSTATE 42601) puesto que el valor del parámetro indica tipos de datos diferentes (REAL o DOUBLE).

Sin embargo, si la longitud, la precisión y la escala están codificadas, el valor debe coincidir exactamente con el que se especifica en la sentencia CREATE PROCEDURE.

No es necesario que un tipo de FLOAT(n) coincida con el valor definido para n puesto que $0 < n < 25$ significa REAL y $24 < n < 54$ significa DOUBLE. La coincidencia se produce basándose en si el tipo es REAL o DOUBLE.

Si no existe ningún procedimiento con la signatura especificada en el esquema nombrado o implícito, se genera un error (SQLSTATE 42883).

SPECIFIC PROCEDURE *nombre-específico*

Indica que se sustituirán o añadirán comentarios para un procedimiento (consulte PROCEDURE para ver otros métodos de identificar un procedimiento). Identifica el procedimiento almacenado en particular que se ha de comentar, utilizando el nombre específico que se ha especificado o que ha tomado por omisión en tiempo creación del procedimiento. En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas,

la opción de precompilación/enlace QUALIFIER especifica de forma implícita el calificador para los nombres de objeto no calificados. El *nombre-especifico* debe identificar una instancia del procedimiento específico en el esquema nombrado o implícito; de lo contrario, se genera un error (SQLSTATE 42704).

El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.PROCEDURES para la fila que describe el procedimiento.

SCHEMA *nombre-esquema*

Indica que se añadirá o sustituirá un comentario para un esquema. El *nombre-esquema* debe identificar un esquema que se describe en el catálogo (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.SCHEMATA para la fila que describe el esquema.

SERVER *nombre-servidor*

Indica que se añadirá o sustituirá un comentario para una fuente de datos. El *nombre-servidor* debe identificar una fuente de datos que está descrita en el catálogo (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.SERVERS correspondiente a la fila que describe la fuente de datos.

SERVER OPTION *nombre-opción-servidor* **FOR** *servidor-remoto*

Indica que se añadirá o se sustituirá un comentario para una opción de servidor.

nombre-opción-servidor

Identifica una opción de servidor. Esta opción ha de ser una que esté descrita en el catálogo (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.SERVEROPTIONS correspondiente a la fila que describe la opción de servidor.

servidor-remoto

Describe la fuente de datos a la que se aplica la *opción-servidor*.

SERVER *nombre-servidor*

Indica el nombre de la fuente de datos a la que se aplica la *opción-servidor*. El *nombre-servidor* debe identificar una fuente de datos que esté descrita en el catálogo.

TYPE *tipo-servidor*

Especifica el tipo de fuente de datos—por ejemplo, DB2 Universal Database para OS/390 u Oracle—al que se aplica la *opción-servidor*. El *tipo-servidor* se puede especificar en minúsculas o mayúsculas; se guardará en mayúsculas en el catálogo.

VERSION

Especifica la versión de la fuente de datos identificada por *nombre-servidor*.

versión

Especifica el número de versión. *versión* debe ser un entero.

release

Especifica el número de release de la versión indicada por *versión*. *release* debe ser un entero.

mod

Especifica el número de la modificación del release indicada por *release*. *mod* debe ser un entero.

constante-serie-versión

Especifica el número completo de la versión. La *constante-serie-versión* puede ser un solo valor (por ejemplo, '8i'); o puede ser los valores concatenados de *versión*, *release* y, si es aplicable, *mod* (por ejemplo, '8.0.3').

WRAPPER *nombre-reiniciador*

Identifica el reiniciador que se utiliza para acceder a la fuente de datos referenciada por *nombre-servidor*.

TABLE *nombre-tabla* o *nombre-vista*

Indica que se añadirá o se sustituirá un comentario para una tabla o vista. El *nombre-tabla* o *nombre-vista* debe identificar una tabla o vista (no un seudónimo ni un apodo) que esté descrita en el catálogo (SQLSTATE 42704) y no debe identificar una tabla temporal declarada (SQLSTATE 42995). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.TABLES correspondiente a la fila que describe la tabla o vista.

TABLESPACE *nombre-espacio-tablas*

Indica que se añadirá o se sustituirá un comentario para un espacio de tablas. El *nombre-espacio-tablas* debe identificar un espacio de tablas diferenciado que esté descrito en el catálogo (SQLSTATE 42704). El comentario sustituye el valor para la columna REMARKS de la vista de catálogo SYSCAT.TABLESPACES para la fila que describe el espacio de tablas.

TRIGGER *nombre-desencadenante*

Indica que se añadirá o se sustituirá un comentario para un desencadenante. El *nombre-desencadenante* debe identificar un desencadenante diferenciado que esté descrito en el catálogo (SQLSTATE 42704). El comentario sustituye el valor para la columna REMARKS de la vista de catálogo SYSCAT.TRIGGERS para la fila que describe el desencadenante.

TYPE *nombre-tipo*

Indica que se añadirá o se sustituirá un comentario para un tipo definido por el usuario. El *nombre-tipo* debe identificar un tipo definido por el usuario que esté descrito en el catálogo (SQLSTATE 42704). Si se especifica **DISTINCT**, *nombre-tipo* debe identificar un tipo diferenciado que esté descrito en el catálogo (SQLSTATE 42704). El comentario sustituye el valor de la columna **REMARKS** de la vista de catálogo **SYSCAT.DATATYPES** para la fila que describe el tipo definido por el usuario.

En las sentencias de SQL dinámicas, el registro especial **CURRENT SCHEMA** se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace **QUALIFIER** especifica implícitamente el calificador para los nombres de objetos no calificados.

TYPE MAPPING *nombre-correlación-tipos*

Indica que se añadirá o sustituirá un comentario para una correlación de tipos de datos definida por el usuario. El *nombre-correlación-tipos* debe identificar una correlación de tipos de datos que esté descrita en el catálogo (SQLSTATE 42704). El comentario sustituye el valor de la columna **REMARKS** de la vista de catálogo **SYSCAT.TYPEMAPPINGS** correspondiente a la fila que describe la correlación.

WRAPPER *nombre-reiniciador*

Indica que se añadirá o sustituirá un comentario para un reiniciador. El *nombre-reiniciador* debe identificar un reiniciador que esté descrito en el catálogo (SQLSTATE 42704). El comentario sustituye el valor de la columna **REMARKS** de la vista de catálogo **SYSCAT.WRAPPERS** correspondiente a la fila que describe el reiniciador.

IS *constante-serie*

Especifica el comentario que va a añadirse o sustituirse. La *constante-serie* puede ser cualquier constante de serie de caracteres con un máximo de 254 bytes. (El retorno de carro y el salto de línea cuentan como 1 byte cada uno.)

*nombre-tabla | nombre-vista ({ nombre-columna **IS** constante-serie } ...)*

Este formato de la sentencia **COMMENT** proporciona la posibilidad de especificar comentarios para múltiples columnas de una tabla o vista. Los nombres de columna no deben estar calificados, cada nombre debe identificar una columna de la vista o tabla especificada y la tabla o vista debe estar descrita en el catálogo. El *nombre-tabla* no puede ser una tabla temporal declarada (SQLSTATE 42995).

No puede realizarse un comentario en una columna de una vista no operativa (SQLSTATE 51024).

Ejemplos

Ejemplo 1: Añada un comentario para la tabla **EMPLOYEE**.

COMMENT

```
COMMENT ON TABLE EMPLOYEE
IS 'Refleja la reorganización del primer trimestre'
```

Ejemplo 2: Añada un comentario para la vista EMP_VIEW1.

```
COMMENT ON TABLE EMP_VIEW1
IS 'Vista de la tabla EMPLOYEE sin información sobre salarios'
```

Ejemplo 3: Añada un comentario para la columna EDLEVEL de la tabla EMPLOYEE.

```
COMMENT ON COLUMN EMPLOYEE.EDLEVEL
IS 'curso más alto aprobado en la escuela'
```

Ejemplo 4: Añada comentarios para dos columnas diferentes de la tabla EMPLOYEE.

```
COMMENT ON EMPLOYEE
(WORKDEPT IS 'vea los nombres en la tabla DEPARTMENT',
EDLEVEL IS 'curso más alto aprobado en la escuela' )
```

Ejemplo 5: Pellow desea realizar un comentario sobre la función CENTRE, que ha creado en su esquema PELLOW, utilizando la signatura para identificar la función específica que se ha de comentar.

```
COMMENT ON FUNCTION CENTRE (INT, FLOAT)
IS 'func CENTRE de Frank, utiliza método Chebychev'
```

Ejemplo 6: McBride desea realizar un comentario sobre otra función CENTRE, que ella creó en el esquema PELLOW, utilizando el nombre específico para identificar la instancia de función que hay que comentar:

```
COMMENT ON SPECIFIC FUNCTION PELLOW.FOCUS92 IS
'La función CENTRE con mayor éxito de Louise, utiliza
la técnica de foco borroso browniana'
```

Ejemplo 7: Realice un comentario sobre la función ATOMIC_WEIGHT en el esquema CHEM, donde se sabe que sólo hay una función con dicho nombre:

```
COMMENT ON FUNCTION CHEM.ATOMIC_WEIGHT
IS 'toma núm. atómico, da peso atómico'
```

Ejemplo 8: Eigler desea realizar un comentario sobre el procedimiento SEARCH, que ha creado en su esquema EIGLER, utilizando la signatura para identificar el procedimiento específico que se ha de comentar.

```
COMMENT ON PROCEDURE SEARCH (CHAR, INT)
IS 'Buscar la masa de Frank y sustituir algoritmo'
```

Ejemplo 9: Macdonald desea realizar un comentario sobre otra función SEARCH, que creó en el esquema EIGLER, utilizando el nombre específico para identificar la instancia de procedimiento que hay que comentar:

```
COMMENT ON SPECIFIC PROCEDURE EIGLER.DESTROY IS
'Buscar masa de Patrick y destruir algoritmo'
```

Ejemplo 10: Realice un comentario sobre la función OSMOSIS en el esquema BIOLOGY, donde se sabe que sólo hay un procedimiento con dicho nombre:

```
COMMENT ON PROCEDURE BIOLOGY.OSMOSIS  
IS 'Modelo de cálculos de ósmosis'
```

Ejemplo 11: Realice un comentario sobre una especificación de índice denominada INDEXSPEC.

```
COMMENT ON INDEX INDEXSPEC  
IS 'Una especificación de índice que indica al optimizador  
que la tabla referenciada por el apodo NICK1 tiene un índice.'
```

Ejemplo 12: Realice un comentario sobre el reiniciador cuyo nombre por omisión es NET8.

```
COMMENT ON WRAPPER NET8  
IS 'El reiniciador de las fuentes de datos asociadas con  
el software cliente Net8 de Oracle.'
```

COMMIT

COMMIT

La sentencia COMMIT termina una unidad de trabajo y confirma los cambios de la base de datos que ha realizado esa unidad de trabajo.

Invocación

Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización

No se necesita.

Sintaxis



Descripción

Finaliza la unidad de trabajo en la que se ejecuta la sentencia COMMIT y se inicia una nueva unidad de trabajo. Se confirman todos los cambios efectuados por las siguientes sentencias durante la unidad de trabajo: ALTER, COMMENT ON, CREATE, DELETE, DROP, GRANT, INSERT, LOCK TABLE, REVOKE, SET INTEGRITY, SET variable-transición y UPDATE.

Sin embargo, las siguientes sentencias no están bajo control de transacciones y los cambios realizados por ellas son independientes de la emisión de la sentencia COMMIT:

- SET CONNECTION,
- SET CURRENT DEFAULT TRANSFORM GROUP
- SET CURRENT DEGREE,
- SET CURRENT EXPLAIN MODE,
- SET CURRENT EXPLAIN SNAPSHOT,
- SET CURRENT PACKAGESET,
- SET CURRENT QUERY OPTIMIZATION,
- SET CURRENT REFRESH AGE,
- SET EVENT MONITOR STATE,
- SET PASSTHRU,
- SET PATH,
- SET SCHEMA,
- SET SERVER OPTION.

Se liberan todos los bloqueos adquiridos por la unidad de trabajo posteriores a su inicio, excepto los bloqueos necesarios para los cursores abiertos que se

han declarado como WITH HOLD. Se cierran todos los cursores abiertos que no están definidos como WITH HOLD. Los cursores abiertos definidos con WITH HOLD permanecen abiertos y el cursor se sitúa antes de la siguiente fila lógica de la tabla resultante.⁶³ Se liberan todos los localizadores de LOB. Observe que esto es verdadero incluso cuando los localizadores están asociados a valores LOB recuperados mediante un cursor que tenga la propiedad WITH HOLD.

Se liberan todos los puntos de salvar definidos dentro de la transacción.

Notas

- Se recomienda encarecidamente que cada proceso de aplicación finalice explícitamente su unidad de trabajo antes de terminar. Si el programa de aplicación finaliza normalmente sin una sentencia COMMIT ni ROLLBACK entonces el gestor de bases de datos intenta una confirmación o retrotracción según el entorno de aplicación. Consulte el manual *Application Development Guide* para la finalización implícita de una transacción en distintos entornos de aplicación.
- Vea “EXECUTE” en la página 1036 para conocer el efecto de COMMIT en las sentencias de SQL dinámicas puestas en antememoria.
- Vea “DECLARE GLOBAL TEMPORARY TABLE” en la página 982 para conocer los posibles efectos de COMMIT en tablas temporales declaradas.

Ejemplo

Confirme las modificaciones en la base de datos efectuadas desde el último punto de confirmación.

```
COMMIT WORK
```

63. Debe ejecutarse una sentencia FETCH antes de emitir una sentencia UPDATE o DELETE posicionada.

SQL compuesto (dinámico)

SQL compuesto (dinámico)

Una sentencia compuesta agrupa otras sentencias en un bloque ejecutable. Se pueden declarar variables de SQL en una sentencia compuesta atómica preparada dinámicamente.

Invocación

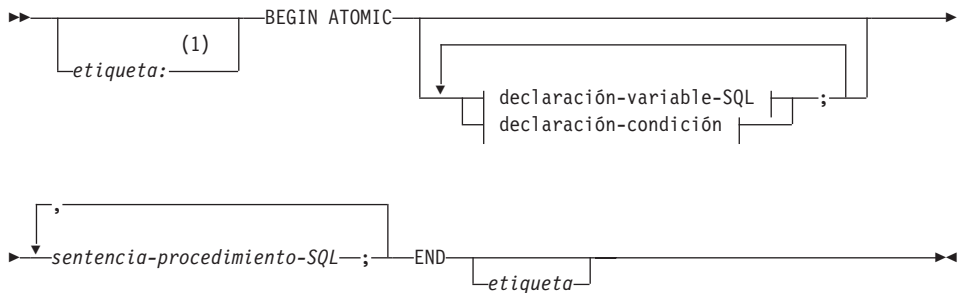
Esta sentencia puede incorporarse en un desencadenante, una función SQL o un método SQL o bien emitirse mediante el uso de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse dinámicamente.

Autorización

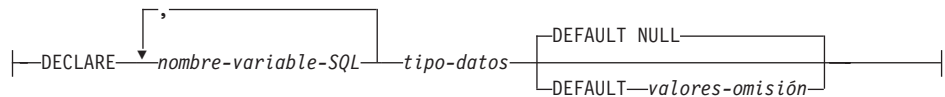
No se necesita ningún privilegio para invocar una sentencia dinámica compuesta. Sin embargo, el ID de autorización de la sentencia compuesta debe tener los privilegios necesarios para invocar las sentencias de SQL incorporadas en la sentencia compuesta.

Sintaxis

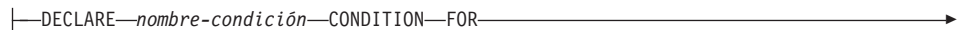
sentencia-dinámica-compuesta

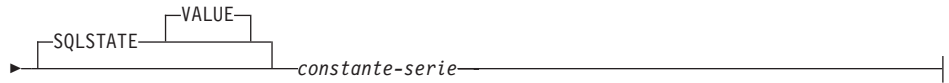


declaración-variable-SQL:



declaración-condición:



**Notas:**

- 1 Sólo se puede especificar una etiqueta cuando la sentencia está en una definición de función, método o desencadenante.

Descripción*etiqueta*

Define la etiqueta del bloque de programa. Si se especifica la etiqueta inicial, ésta puede utilizarse para calificar las variables SQL declaradas en la sentencia dinámica compuesta y también se puede especificar en una sentencia LEAVE. Si se especifica la etiqueta final, ésta deberá ser igual que la etiqueta inicial.

ATOMIC

ATOMIC indica que, si se produce un error en la sentencia compuesta, todas las sentencias de SQL de la sentencia compuesta se retrotraerán y las sentencias de SQL restantes de la sentencia compuesta no se procesarán.

sentencia-procedimiento-SQL

Se puede utilizar la lista siguiente de *sentencias-control-SQL* en la sentencia dinámica compuesta:

- Sentencia FOR
- Sentencia GET DIAGNOSTICS
- Sentencia IF
- Sentencia ITERATE
- Sentencia LEAVE
- Sentencia SIGNAL
- Sentencia WHILE

Las sentencias de SQL que se pueden emitir son:

- fullselect⁶⁴
- UPDATE de búsqueda
- DELETE de búsqueda
- INSERT
- Sentencia variable SET

64. Puede preceder a fullselect una expresión-tabla-común

SQL compuesto (dinámico)

declaración-variable-SQL

Declara una variable que es local en la sentencia dinámica compuesta.

nombre-variable-SQL

Define el nombre de una variable local. DB2 convierte a mayúsculas todos los nombres de variables del SQL. El nombre no puede:

- ser igual que el de otra variable de SQL de la misma sentencia compuesta
- ser igual que un nombre de parámetro
- ser igual que los nombres de columna.

Si una sentencia de SQL contiene un identificador que tiene el mismo nombre que una variable SQL y una referencia a columna, DB2 interpreta el identificador como una columna.

tipo-datos

Especifica el tipo de datos de la variable.

DEFAULT *valores-omisión* o NULL

Define el valor por omisión de la variable SQL. La variable se inicializa al llamar a la sentencia dinámica compuesta. Si no se especifica un valor por omisión, el valor de la variable se inicializa en NULL.

declaración-condición

Declara el nombre de una condición y el valor SQLSTATE asociado.

nombre-condición

Especifica el nombre de la condición. El nombre de la condición debe ser exclusivo dentro del cuerpo del procedimiento y sólo puede estar referenciado dentro de la sentencia compuesta donde está declarado.

FOR SQLSTATE *constante-serie*

Especifica el SQLSTATE asociado con la condición. La *constante-serie* debe especificarse en forma de cinco caracteres entre comillas simples y no se puede ser '00000'.

Notas

- DB2 compila las sentencias dinámicas compuestas como una sola sentencia. Esta sentencia está en vigor para scripts cortos que incluyan poca lógica de flujo de control pero un flujo de datos significativo. Para construcciones mayores con un flujo de control complejo anidado, los procedimientos SQL son una elección mejor. Consulte el apartado "CREATE PROCEDURE" en la página 804 para obtener más detalles sobre la utilización de los procedimientos SQL.

Ejemplos

Ejemplo 1:

Este ejemplo ilustra cómo se puede utilizar PL SQL incorporado en un escenario de depósito de datos para la limpieza de datos.

El ejemplo presenta tres tablas. La tabla "target" contiene los datos limpiados. La tabla "except" almacena filas que no se pueden limpiar (excepciones) y la tabla "source" contiene los datos sin procesar que se deben limpiar.

Se utiliza una función SQL simple llamada "discretize" para clasificar y modificar los datos. Ésta devuelve NULL para todos los datos incorrectos. Entonces la sentencia dinámica compuesta limpia los datos. Recorre todas las filas de la tabla fuente en un bucle-FOR y decide si la fila actual se inserta en la tabla "target" o la tabla "except", dependiendo del resultado de la función "discretize". Con esta técnica es posible utilizar mecanismos más elaborados (limpieza en varias etapas).

Se puede escribir el mismo código utilizando un procedimiento SQL o cualquier otro procedimiento o aplicación en un lenguaje principal. Sin embargo, la sentencia dinámica compuesta ofrece una ventaja exclusiva porque el bucle-FOR no abre un cursor y las inserciones de fila individuales no son realmente inserciones de fila individuales. De hecho, la lógica es en efecto una inserción de múltiples tablas desde una selección compartida.

Esto se logra mediante la compilación de la sentencia dinámica compuesta como una sentencia individual. De forma similar a una vista cuyo cuerpo se integra en la consulta que la utiliza y entonces se compila y se optimiza como una totalidad en el contexto de consulta, el optimizador DB2 compila y optimiza el flujo de control y el flujo de datos. Por consiguiente, la lógica entera se ejecuta en tiempo de ejecución de DB2'. No se mueven datos fuera del motor central de DB2, como sucedería en el caso de un procedimiento almacenado.

El primer paso es crear las tablas necesarias:

```
CREATE TABLE target
  (pk INTEGER
  NOT NULL          PRIMARY KEY, c1 INTEGER)
```

Esto crea una tabla llamada "target" que va a contener los datos limpiados.

```
CREATE TABLE except
  (pk INTEGER
  NOT NULL          PRIMARY KEY, c1 INTEGER)
```

Esto crea una tabla llamada "except" que va a contener las excepciones.

```
CREATE TABLE source
  (pk INTEGER
  NOT NULL          PRIMARY KEY, c1 INTEGER)
```

SQL compuesto (dinámico)

Esto crea una tabla llamada "source" que contiene los datos que se deben limpiar.

A continuación, creamos una función "discretize" para limpiar los datos echando fuera todos los valores [0..1000] y alineándolos en pasos de 10.

```
CREATE FUNCTION Discretize(row INTEGER) RETURNS INTEGER
  RETURN CASE
    WHEN row < 0 THEN CAST(NULL AS INTEGER)
    WHEN row > 1000 THEN NULL
    ELSE ((row / 10) * 10) + 5
  END

BEGIN ATOMIC
  FOR row AS
    SELECT pk, c1, Discretize(c1) AS d FROM source
DO
  IF row.d is NULL THEN
    INSERT INTO except VALUES(row.pk, row.c1);
ELSE
  INSERT INTO target VALUES(row.pk, row.d);
END IF;
END FOR;
END
```

Entonces podemos insertar los valores:

```
INSERT INTO source (pk, c1) VALUES
(1, -5),
(2, NULL),
(3, 1200),
(4, 23),
(5, 10),
(6, 876)
```

Y comprobar los resultados:

```
SELECT * FROM except ORDER BY 1
```

PK	C1
1	-5
2	-
3	1200

3 registros(s) seleccionado(s).

```
SELECT * FROM target ORDER BY 1
```

PK	C1
4	25
5	15
6	875

3 registro(s) seleccionado(s).

El paso final consiste en limpiar:

```
DROP FUNCTION Discretize  
DROP TABLE source  
DROP TABLE target  
DROP TABLE except
```

SQL compuesto (incorporado)

SQL compuesto (incorporado)

Combina una o varias sentencias de SQL (*subsencencias*) en un bloque ejecutable. Consulte “Capítulo 7. Sentencias de control de SQL” en la página 1209 para ver sentencias de SQL compuestas dentro de procedimientos SQL.

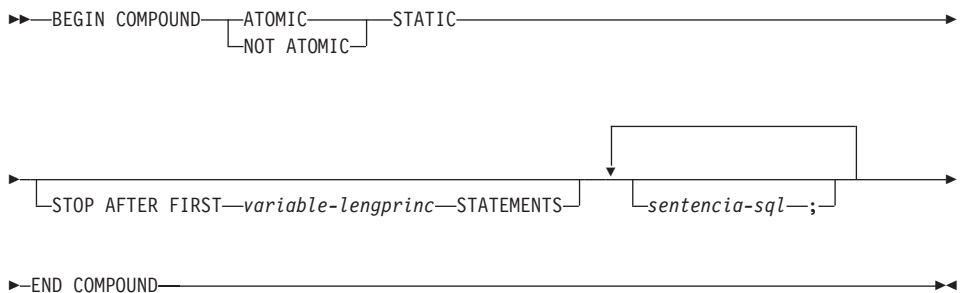
Invocación

Esta sentencia sólo puede incluirse en un programa de aplicación. Toda la construcción de la sentencia de SQL compuesta es una sentencia ejecutable que no puede prepararse dinámicamente. La sentencia no está soportada en REXX.

Autorización

Ninguna para la sentencia de SQL compuesta propiamente dicha. El ID de autorización de la sentencia de SQL compuesta debe tener la autorización adecuada en todas las sentencias individuales que están contenidas dentro de la sentencia de SQL compuesta.

Sintaxis



Descripción

ATOMIC

Especifica que, si falla alguna de las subsencencias de la sentencia de SQL compuesta, se deshacen todos los cambios efectuados en la base de datos por cualquiera de las subsencencias, incluidos los cambios realizados por subsencencias satisfactorias.

NOT ATOMIC

Especifica que, sin tener en cuenta si falla alguna subsencencia, la sentencia de SQL compuesta no desharrá ningún cambio efectuado en la base de datos por las otras subsencencias.

STATIC

Especifica que las variables de entrada para todas las subsencencias conservan su valor original. Por ejemplo, si

```
SELECT ... INTO :abc ...
```

va seguido de:

```
UPDATE T1 SET C1 = 5 WHERE C2 = :abc
```

la sentencia UPDATE utilizará el valor que :abc tenía al principio de la ejecución de la sentencia de SQL compuesta, no el valor que sigue a SELECT INTO.

Si más de una subsentencia establece la misma variable, el valor de dicha variable después de la sentencia de SQL compuesta es el valor establecido por la última subsentencia.

Nota: No se da soporte al funcionamiento no estático. Esto quiere decir que la ejecución de las subsentencias no es secuencial y que no deben tener interdependencias.

STOP AFTER FIRST

Especifica que sólo se ejecutarán un número determinado de subsentencias.

variable-lengprinc

Un entero pequeño que especifica el número de subsentencias que se deben ejecutar.

STATEMENTS

Completa la cláusula STOP AFTER FIRST *variable-lengprinc*.

sentencia-sql

Todas las sentencias ejecutables, excepto las siguientes, pueden estar contenidas en una sentencia de SQL compuesta estática incorporada:

CALL	OPEN
CLOSE	PREPARE
CONNECT	RELEASE (Conexión)
SQL compuesto	RELEASE SAVEPOINT
DESCRIBE	ROLLBACK
DISCONNECT	SAVEPOINT
EXECUTE IMMEDIATE	SET CONNECTION
FETCH	

Si se incluye una sentencia COMMIT, debe ser la última subsentencia. Si COMMIT está en esta posición, se emitirá aunque la cláusula STOP AFTER FIRST *variable-lengprinc* STATEMENTS indique que no deben ejecutarse todas las subsentencias. Por ejemplo, suponga que COMMIT es la última subsentencia en un bloque SQL compuesto formado por 100

SQL compuesto (incorporado)

subsentencias. Si la cláusula STOP AFTER FIRST STATEMENTS indica que sólo deben ejecutarse 50 subsentencias, entonces COMMIT será la subsentencia número 51.

Se devolverá un error si se incluye COMMIT al utilizar CONNECT TYPE 2 o ejecutar en un entorno de proceso de transacciones distribuidas XA (SQLSTATE 25000).

Normas

- DB2 Connect no da soporte a las sentencias SELECT que seleccionan columnas LOB en un bloque de SQL compuesto.
- No está permitido ningún código de lenguaje principal en una sentencia de SQL compuesta; es decir, no está permitido ningún código de lenguaje principal entre las subsentencias que componen la sentencia de SQL compuesta.
- DB2 Connect sólo acepta sentencias de SQL compuestas no atómicas (NOT ATOMIC).
- Las sentencias de SQL compuestas no pueden anidarse.
- No se puede emitir una sentencia de SQL compuesta atómica dentro de un punto de salvar (SQLSTATE 3B002).
- No se permite una sentencia COMMIT preparada en una sentencia de SQL compuesta ATOMIC

Notas

Se devuelve una SQLCA para toda la sentencia de SQL compuesta. La mayor parte de la información de dicha SQLCA refleja los valores establecidos por el servidor de aplicaciones cuando ha procesado la última subsentencia. Por ejemplo:

- Normalmente, SQLCODE y SQLSTATE son los que corresponden a la última subsentencia (la excepción se describe en el punto siguiente).
- Si se devuelve un aviso de 'no se han encontrado datos' (SQLSTATE '02000'), ese aviso tiene prioridad sobre cualquier otro aviso para que se pueda responder a una excepción WHENEVER NOT FOUND.⁶⁵
- Los indicadores SQLWARN son una acumulación de los indicadores establecidos para todas las subsentencias.

Si se han producido uno o más errores durante la ejecución de NOT ATOMIC del SQL compuesto y ninguno de ellos es grave, SQLERRMC contendrá información sobre un máximo de siete errores. El primer símbolo de

65. Esto significa que los campos SQLCODE, SQLERRML, SQLERRMC y SQLERRP de la SQLCA que finalmente se devuelve a la aplicación son los campos de la subsentencia que desencadenó el aviso de 'no se han encontrado datos'. Si hay más de un aviso 'no se han encontrado datos' en la sentencia de SQL compuesta, los campos devueltos serán los correspondientes a la última subsentencia.

SQLERRMC indicará el número total de errores que se han producido. Los símbolos restantes contendrán cada uno la posición de orden y el SQLSTATE de la subsentencia anómala dentro de la sentencia de SQL compuesta. El formato es una serie de caracteres en el formato:

nnnXssscccc

en la que la subserie que empieza por X se repite hasta seis veces más y los elementos de la serie están definidos de la manera siguiente.

- nnn** El número total de sentencias que han producido errores. ⁶⁶ Este campo se justifica por la izquierda y se rellena con blancos.
- X** El separador de símbolos X'FF'.
- sss** La posición ordinal de la sentencia que ha provocado el error. ⁶⁶ Por ejemplo, si la primera sentencia ha fallado, este campo contendría el número uno justificado por la izquierda ('1 ').
- cccc** El SQLSTATE del error.

El segundo campo SQLERRD contiene el número de sentencias que han fallado (han devuelto SQLCODE negativos).

El tercer campo SQLERRD de la SQLCA es una acumulación del número de filas afectadas por todas las subsentencias.

El cuarto campo SQLERRD de la SQLCA es una cuenta del número de subsentencias satisfactorias. Si, por ejemplo, falla la tercera subsentencia de una sentencia de SQL compuesta, el cuarto campo SQLERRD se establecería en 2, lo que indicaría que 2 subsentencias se habrían procesado de manera satisfactoria antes de encontrar el error.

El quinto campo SQLERRD de la SQLCA es una acumulación del número de filas actualizadas o suprimidas a causa de imposición de las restricciones de integridad de referencia para todas las subsentencias que han activado dicha actividad de restricción.

Ejemplos

Ejemplo 1: En un programa C, emita una sentencia de SQL compuesto que actualice las tablas ACCOUNTS y TELLERS. Si hay un error en cualquiera de las sentencias, deshaga el efecto de todas las sentencias (ATOMIC). Si no hay errores, confirme la unidad de trabajo actual.

```
EXEC SQL BEGIN COMPOUND ATOMIC STATIC
      UPDATE ACCOUNTS SET ABALANCE = ABALANCE + :delta
      WHERE AID = :aid;
      UPDATE TELLERS SET TBALANCE = TBALANCE + :delta
```

66. Si el número excede de 999, la cuenta vuelve a empezar en cero.

SQL compuesto (incorporado)

```
WHERE TID = :tid;  
INSERT INTO TELLERS (TID, BID, TBALANCE) VALUES (:i, :branch_id, 0);  
COMMIT;  
END COMPOUND;
```

Ejemplo 2: En un programa C, inserte 10 filas de datos en la base de datos Suponga que la variable del lenguaje principal :nbr contiene el valor 10 y S1 es una sentencia INSERT preparada. Además, suponga que todas las inserciones deben intentarse sin tener en cuenta los errores (NOT ATOMIC).

```
EXEC SQL BEGIN COMPOUND NOT ATOMIC STATIC STOP AFTER FIRST :nbr STATEMENTS  
EXECUTE S1 USING DESCRIPTOR :*sqlda0;  
EXECUTE S1 USING DESCRIPTOR :*sqlda1;  
EXECUTE S1 USING DESCRIPTOR :*sqlda2;  
EXECUTE S1 USING DESCRIPTOR :*sqlda3;  
EXECUTE S1 USING DESCRIPTOR :*sqlda4;  
EXECUTE S1 USING DESCRIPTOR :*sqlda5;  
EXECUTE S1 USING DESCRIPTOR :*sqlda6;  
EXECUTE S1 USING DESCRIPTOR :*sqlda7;  
EXECUTE S1 USING DESCRIPTOR :*sqlda8;  
EXECUTE S1 USING DESCRIPTOR :*sqlda9;  
END COMPOUND;
```

CONNECT (Tipo 1)

La sentencia CONNECT (Tipo 1) conecta un proceso de aplicación con el servidor de aplicaciones identificado según las reglas para una unidad de trabajo remota.

Un proceso de aplicación sólo puede estar conectado a un único servidor de aplicaciones en un momento dado. Se denomina *servidor actual*. Puede establecerse un servidor de aplicaciones por omisión cuando se inicializa el peticionario de aplicaciones. Si la conexión implícita está disponible y se inicia un proceso de aplicación, se conecta de manera implícita al servidor de aplicaciones por omisión. El proceso de aplicación puede conectarse explícitamente a un servidor de aplicaciones distinto emitiendo una sentencia CONNECT TO. La conexión dura hasta que se emite una sentencia CONNECT RESET o una sentencia DISCONNECT o hasta que otra sentencia CONNECT TO cambia el servidor de aplicaciones.

Consulte el apartado “Gestión de conexión de la unidad de trabajo remota” en la página 45 para conocer conceptos y detalles adicionales acerca de los estados de conexión. Consulte el apartado “Opciones que controlan la semántica de la unidad de trabajo distribuida” en la página 54 para saber las opciones de precompilador que determinan la infraestructura para el funcionamiento de CONNECT.

Invocación

Aunque un recurso SQL interactivo pueda proporcionar una interfaz que dé la apariencia de ejecución interactiva, esta sentencia sólo puede incluirse dentro de un programa de aplicación. Se trata de una sentencia ejecutable que no puede prepararse de forma dinámica.

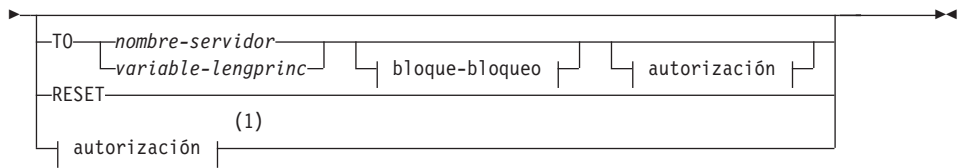
Autorización

El ID de autorización de la sentencia debe estar autorizado para conectar con el servidor de aplicaciones identificado. Según cual sea el valor de autenticación definido para la base de datos, el control de autorizaciones puede realizarlo el cliente o el servidor. En una base de datos particionada, las definiciones de usuario y de grupo deben ser idénticas en todas las particiones o nodos. Consulte el parámetro de configuración AUTHENTICATION del gestor de bases de datos en el manual *Administration Guide* para obtener información acerca del valor de la autenticación.

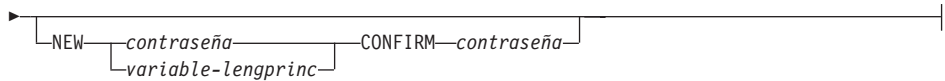
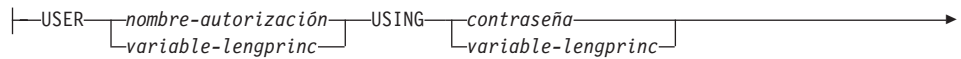
Sintaxis

►►—CONNECT—►

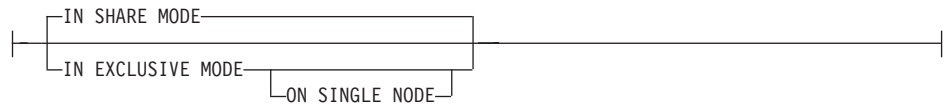
CONNECT (Tipo 1)



autorización:



bloque-bloqueo:



Notas:

- 1 Este formato sólo es válido si se ha habilitado la conexión implícita.

Descripción

CONNECT (sin ningún operando)

Devuelve información acerca del servidor actual. La información se devuelve en el campo `SQLERRP` de la `SQLCA`, tal como se describe en "Conexión satisfactoria".

Si existe un estado de conexión, el ID de autorización y el seudónimo de base de datos se colocan en el campo `SQLERRMC` de la `SQLCA`. Si el ID de autorización es mayor que 8 bytes, se truncará a 8 bytes, y el truncamiento se indicará en los campos `SQLWARN0` y `SQLWARN1` de la `SQLCA`, mediante 'W' y 'A', respectivamente. Si el parámetro de configuración `DYN_QUERY_MGMT` de la base de datos está habilitado, los campos `SQLWARN0` y `SQLWARN7` de la `SQLCA` se marcarán con los indicadores 'W' y 'E', respectivamente.

Si no existe ninguna conexión y es posible la conexión implícita, se intenta efectuar una conexión implícita. Si no hay una conexión implícita disponible, este intento produce un error (no hay ninguna conexión existente). Si no hay conexión, el campo `SQLERRMC` está en blanco.

El código de país y la página de códigos del servidor de aplicaciones se colocan en el campo SQLERRMC (como en el caso de una sentencia CONNECT TO satisfactoria).

Esta forma de CONNECT:

- No necesita que el proceso de aplicación esté en estado conectable.
- Si está conectado, no cambia el estado de conexión.
- Si está desconectado y está disponible la conexión implícita, se realiza una conexión con el servidor de aplicaciones por omisión. En este caso, el código de país y la página de códigos del servidor de aplicaciones se colocan en el campo SQLERRMC, como una sentencia CONNECT TO satisfactoria.
- Si está desconectado y no está disponible la conexión implícita, el proceso de aplicación permanece desconectado.
- No cierra los cursores.

TO *nombre-servidor* o *variable-lengprinc*

Identifica el servidor de aplicaciones mediante el *nombre-servidor* especificado o una *variable-lengprinc* que contenga el nombre-servidor.

Si se especifica una *variable-lengprinc*, debe ser una variable de serie de caracteres con un atributo de longitud que no sea mayor que 8, y no debe contener una variable indicadora. El *nombre-servidor* que está contenido dentro de la *variable-lengprinc* debe estar justificado por la izquierda y no estar encerrado entre comillas.

Observe que el *nombre-servidor* es un seudónimo de base de datos que identifica al servidor de aplicaciones. Debe estar listado en el directorio local del petionario de aplicaciones.

Nota: DB2 para MVS da soporte a un nombre de ubicación de 16 bytes y tanto SQL/DS como DB2/400 dan soporte a un nombre de base de datos de destino de 18 bytes. DB2 Versión 7 sólo da soporte a la utilización de un seudónimo de base de datos de 8 bytes en la sentencia CONNECT. No obstante, el seudónimo de base de datos puede correlacionarse con un nombre de base de datos de 18 bytes a través del Directorio de servicio de conexiones a bases de datos.

Cuando se ejecuta la sentencia CONNECT TO, el proceso de aplicación debe estar en estado conectable (consulte el apartado “Gestión de conexión de la unidad de trabajo remota” en la página 45 para obtener información acerca de los estados de conexión con CONNECT Tipo 1).

Conexión satisfactoria:

Si la sentencia CONNECT TO es satisfactoria:

CONNECT (Tipo 1)

- Se cierran todos los cursores abiertos, se destruyen todas las sentencias preparadas y se liberan todos los bloqueos del servidor de aplicaciones anterior.
- El proceso de aplicación se desconecta de su servidor de aplicaciones anterior, si lo hay, y se conecta al servidor de aplicaciones identificado.
- El nombre real del servidor de aplicaciones (no un seudónimo) se coloca en el registro especial CURRENT SERVER.
- Se coloca información acerca del servidor de aplicaciones en el campo SQLERRP de SQLCA. Si el servidor de aplicaciones es un producto IBM, la información tiene el formato *pppvrrm*, donde:
 - *ppp* identifica el producto de la manera siguiente:
 - DSN para DB2 para MVS
 - ARI para SQL/DS
 - QSQ para DB2/400
 - SQL para DB2 Universal Database
 - *vv* es un identificador de versión de dos dígitos como, por ejemplo, '02'
 - *rr* es un identificador de release de dos dígitos como, por ejemplo, '01'
 - *m* es un identificador de nivel de modificación de un dígito como, por ejemplo, '0'.

Por ejemplo, si el servidor de aplicaciones es de la Versión 1 Release 1 de DB2 Universal Database para OS/2, el valor de SQLERRP es 'SQL01010'.⁶⁷

- El campo SQLERRMC de la SQLCA se establece en los valores siguientes (separados por X'FF')
1. el código de país del servidor de aplicaciones (o blancos si utiliza DDCS),
 2. la página de códigos del servidor de aplicaciones (o CCSID si utiliza DDCS),
 3. El ID de autorización (sólo los primeros 8 bytes como máximo),
 4. el seudónimo de base de datos,
 5. el tipo de plataforma del servidor de aplicaciones. Los valores reconocidos actualmente son:

Símbolo	Servidor
QAS	DB2 Universal Database para AS/400

67. Este release de DB2 Universal Database Versión 7 es 'SQL07010'.

QDB2	DB2 Universal Database para OS/390
QDB2/2	DB2 Universal Database para OS/2
QDB2/6000	DB2 Universal Database para AIX
QDB2/HPUX	DB2 Universal Database para HP-UX
QDB2/LINUX	DB2 Universal Database para Linux
QDB2/NT	DB2 Universal Database para Windows NT
QDB2/PTX	DB2 Universal Database para NUMA-Q
QDB2/SCO	DB2 Universal Database para SCO UnixWare
QDB2/SNI	DB2 Universal Database para Siemens Nixdorf
QDB2/SUN	DB2 Universal Database para el sistema operativo Solaris
QDB2/Windows 95	DB2 Universal Database para Windows 95 o Windows 98
QSQLDS/VM	DB2 Server para VM
QSQLDS/VSE	DB2 Server para VSE

6. El ID de agente. Identifica al agente que se ejecuta en el gestor de bases de datos en nombre de la aplicación. Este campo es el mismo que el elemento `id_agente` devuelto por el supervisor de bases de datos.
7. El índice de agente. Identifica el índice del agente y se utiliza para el servicio.
8. El número de partición. Para una base de datos no particionada, siempre es 0, si está presente.
9. La página de códigos de la aplicación cliente.
10. El número de particiones de una base de datos particionada. Si la base de datos no puede particionarse, el valor es 0 (cero). El símbolo sólo está presente con la Versión 5 o posterior.

- El campo `SQLERRD(1)` de la `SQLCA` indica la diferencia máxima esperada en la longitud de los datos de caracteres mixtos (tipos de

68. Consulte la sección "Character Conversion Expansion Factor" del capítulo "Programming in Complex Environments" del manual *Application Development Guide* para ver los detalles.

CONNECT (Tipo 1)

datos CHAR) al convertirlos a la página de códigos de la base de datos a partir de la página de códigos de la aplicación. El valor 0 ó 1 indica que no se ha producido expansión; un valor mayor que 1 indica una posible expansión en la longitud; un valor negativo indica una posible compresión.⁶⁸

- El campo SQLERRD(2) de la SQLCA indica la diferencia máxima esperada en la longitud de los datos de caracteres mixtos (tipos de datos CHAR) al convertirlos a la página de códigos de la aplicación a partir de la página de códigos de la base de datos. El valor 0 ó 1 indica que no se ha producido expansión; un valor mayor que 1 indica una posible expansión en la longitud; un valor negativo indica una posible compresión.⁶⁸
- El campo SQLERRD(3) de la SQLCA indica si la base de datos de la conexión es actualizable o no. Inicialmente, una base de datos es actualizable, pero se cambia por de sólo lectura si una unidad de trabajo determina que el ID de autorización no puede efectuar actualizaciones. El valor es uno de los siguientes:
 - 1 - actualizable
 - 2 - sólo lectura
- El campo SQLERRD(4) de la SQLCA devuelve ciertas características de la conexión. El valor es uno de los siguientes:
 - 0 - N/D (sólo es posible si se ejecuta desde un cliente de nivel inferior que tenga una confirmación de una fase y sea un actualizador).
 - 1 - confirmación de una fase.
 - 2 - confirmación de una fase; sólo lectura (únicamente aplicable a las conexiones con bases de datos DRDA1 en un entorno de Supervisor TP).
 - 3 - confirmación de dos fases.
- El campo SQLERRD(5) de la SQLCA devuelve el tipo de autenticación de la conexión. El valor es uno de los siguientes:
 - 0 - Autenticado en el servidor.
 - 1 - Autenticado en el cliente.
 - 2 - Autenticado utilizando DB2 Connect.
 - 3 - Autenticado utilizando los servicios de seguridad Distributed Computing Environment.
 - 255 - Autenticación no especificada.

Consulte el apartado "Controlling Database Access" del manual *Administration Guide* para ver los detalles sobre los tipos de autenticación.

- El campo SQLERRD(6) de SQLCA devuelve el número de partición para la que se ha realizado la conexión si la base de datos está particionada. De lo contrario, se devuelve un valor de 0.

- El campo SQLWARN1 de la SQLCA se establecerá en 'A' si el ID de autorización de la conexión satisfactoria es mayor que 8 bytes. Esto indica que se ha producido un truncamiento. El campo SQLWARN0 de la SQLCA se establecerá en 'W' para indicar este aviso.
- El campo SQLWARN7 de la SQLCA se establecerá en 'E' si el parámetro de configuración DYN_QUERY_MGMT de la base de datos está habilitado. El campo SQLWARN0 de la SQLCA se establecerá en 'W' para indicar este aviso.

Conexión no satisfactoria:

Si la sentencia CONNECT TO no es satisfactoria:

- El campo SQLERRP de la SQLCA se establece en el nombre del módulo del peticionario de aplicaciones que ha detectado el error. Observe que los tres primeros caracteres del nombre de módulo identifican el producto. Por ejemplo, si el peticionario de aplicaciones se encuentra en el gestor de bases de datos del OS/2, los tres primeros caracteres son 'SQL'.
- Si la sentencia CONNECT TO no es satisfactoria porque el proceso de aplicación no se encuentra en estado conectable, el estado de conexión del proceso de aplicación no se modifica.
- Si la sentencia CONNECT TO no es satisfactoria porque el *nombre-servidor* no aparece listado en el directorio local, se emite un mensaje de error (SQLSTATE 08001) y el estado de conexión del proceso de aplicación no se modifica:
 - Si el peticionario de aplicación no estaba conectado a un servidor de aplicaciones, el proceso de aplicación sigue sin estar conectado.
 - Si el peticionario de aplicaciones ya estaba conectado a un servidor de aplicaciones, el proceso de aplicación permanece conectado a ese servidor de aplicaciones. Las sentencias posteriores se ejecutan en dicho servidor de aplicaciones.
- Si la sentencia CONNECT TO no es satisfactoria por cualquier otra razón, el proceso de aplicación se coloca en estado no conectado.

IN SHARE MODE

Permite otras conexiones simultáneas con la base de datos e impide que otros usuarios se conecten a la base de datos en modalidad exclusiva.

IN EXCLUSIVE MODE⁶⁹

Evita que los procesos de aplicación simultáneos ejecuten cualquier operación en el servidor de aplicaciones, a menos que tengan el mismo ID de autorización que el usuario que mantiene el bloqueo exclusivo.

69. DDCS no da soporte a esta opción.

CONNECT (Tipo 1)

ON SINGLE NODE

Especifica que la partición coordinadora está conectada en modalidad exclusiva y que todas las demás particiones están conectadas en modalidad de compartimiento. Esta opción sólo es efectiva en una base de datos particionada.

RESET

Desconecta el proceso de aplicación del servidor actual. Se realiza una operación de confirmación. Si no está disponible la conexión implícita, el proceso de aplicación continúa no conectado hasta que se emite una sentencia de SQL.

USER *nombre-autorización/variable-lengprinc*

Identifica el id de usuario que intenta conectarse al servidor de aplicaciones. Si se especifica una variable del lenguaje principal, debe ser una variable de serie de caracteres con un atributo de longitud no superior a 8 y no debe contener ninguna variable indicadora. El ID de usuario contenido en la *variable-lengprinc* debe estar justificado por la izquierda y no debe estar delimitado por comillas.

USING *contraseña/variable-lengprinc*

Identifica la contraseña del ID de usuario que intenta conectarse al servidor de aplicaciones. La *contraseña* o *variable-lengprinc* pueden tener hasta 18 caracteres como máximo. Si se especifica una variable del lenguaje principal, debe ser una variable de serie de caracteres con un atributo de longitud no superior a 18 y no debe incluir una variable indicadora.

NEW *contraseña/variable-lengprinc* CONFIRM *contraseña*

Identifica la nueva contraseña que debe asignarse al id de usuario identificado por la opción USER. La *contraseña* o *variable-lengprinc* pueden tener hasta 18 caracteres como máximo. Si se especifica una variable del lenguaje principal, debe ser una variable de serie de caracteres con un atributo de longitud no superior a 18 y no debe incluir una variable indicadora. El sistema en que se cambiará la contraseña depende de cómo esté configurada la autenticación de usuario.

Notas

- Es recomendable que la primera sentencia de SQL que se ejecute en un proceso de aplicación sea la sentencia CONNECT TO.
- Si se emite una sentencia CONNECT TO para el servidor de aplicaciones actual con un ID de usuario y una contraseña diferentes, la conversación se desasigna y se vuelve a asignar. El gestor de bases de datos cierra todos los cursores (con pérdida de la posición del cursor si se ha utilizado la opción WITH HOLD).
- Si se emite una sentencia CONNECT TO para el servidor de aplicaciones actual con el mismo ID de usuario y contraseña, la conversación no se desasigna ni se vuelve a asignar. En este caso, los cursores no se cierran.

- Para utilizar DB2 Universal Database Enterprise - Extended Edition, el usuario o la aplicación deben conectarse a una de las particiones listadas en el archivo `db2nodes.cfg` (consulte el apartado “Partición de datos entre múltiples particiones” en la página 40 para obtener información acerca de este archivo). Debe intentar asegurar que no todos los usuarios utilicen la misma partición como partición coordinadora.

Ejemplos

Ejemplo 1: En un programa C, conéctese al servidor de aplicaciones TOROLAB3, donde TOROLAB3 es un seudónimo de la base de datos del mismo nombre, con el id de usuario FERMAT y la contraseña THEOREM.

```
EXEC SQL CONNECT TO TOROLAB3 USER FERMAT USING THEOREM;
```

Ejemplo 2: En un programa C, conéctese a un servidor de aplicaciones cuyo seudónimo de base de datos esté almacenado en la variable del lenguaje principal APP_SERVER (varchar(8)). Tras una conexión satisfactoria, copie el identificador de producto de tres caracteres del servidor de aplicaciones a la variable PRODUCT (char(3)).

```
EXEC SQL CONNECT TO :APP_SERVER;  
if (strcmp(SQLSTATE,'00000',5))  
    strncpy(PRODUCT,sqlca.sqlerrp,3);
```

CONNECT (Tipo 2)

CONNECT (Tipo 2)

La sentencia CONNECT (Tipo 2) conecta un proceso de aplicación con el servidor de aplicaciones identificado y establece las reglas para una unidad de trabajo distribuida y dirigida por aplicación. Este servidor es entonces el servidor actual para el proceso.

Consulte el apartado “Unidad de trabajo distribuida dirigida por aplicación” en la página 49 para conocer conceptos y detalles adicionales.

La mayor parte de los aspectos de una sentencia CONNECT (Tipo 1) son también aplicables a una sentencia CONNECT (Tipo 2). En lugar de repetir ahora lo mismo, esta sección sólo describe los elementos del Tipo 2 que difieren del Tipo 1.

Invocación

La invocación es la misma que la del apartado “Invocación” en la página 653.

Autorización

La autorización es la misma la del apartado “Autorización” en la página 653.

Sintaxis

La sintaxis es la misma que la del apartado “Sintaxis” en la página 653. La selección entre el Tipo 1 y el Tipo 2 se determina según las opciones de precompilador. Encontrará una visión general de estas opciones en el apartado “Opciones que controlan la semántica de la unidad de trabajo distribuida” en la página 54. Se proporcionan más detalles en los manuales *Consulta de mandatos* y *Administrative API Reference*.

Descripción

TO *nombre-servidor/variable-lengprinc*

Las reglas para codificar el nombre del servidor son las mismas que para el Tipo 1.

Si la opción SQLRULES(STD) está en vigor, el *nombre-servidor* no debe identificar una conexión existente del proceso de aplicación; de lo contrario, se produce un error (SQLSTATE 08002).

Si la opción SQLRULES(DB2) está en vigor y el *nombre-servidor* identifica una conexión existente del proceso de aplicación, esa conexión se convierte en la actual y la conexión anterior se coloca en estado inactivo. Es decir, el efecto de la sentencia CONNECT en esta situación es el mismo que el de la sentencia SET CONNECTION.

Consulte el apartado “Opciones que controlan la semántica de la unidad de trabajo distribuida” en la página 54 para obtener información acerca de la especificación de SQLRULES.

Conexión satisfactoria

Si la sentencia CONNECT TO es satisfactoria:

- Se crea (o deja de estar inactiva) una conexión con el servidor de aplicaciones y se coloca en estado actual y mantenido.
- Si CONNECT TO se dirige a un servidor diferente del actual, la conexión actual se coloca en estado inactivo.
- El registro especial CURRENT SERVER y la SQLCA se actualizan de la misma manera que para la CONNECT Tipo 1; vaya a la página 655.

Conexión no satisfactoria

Si la sentencia CONNECT TO no es satisfactoria:

- No importa cuál haya sido la razón de la anomalía, el estado de conexión del proceso de aplicación y los estados de sus conexiones permanecen invariables.
- Al igual que para un CONNECT de Tipo 1 no satisfactorio, el campo SQLERRP de la SQLCA se establece en el nombre del módulo en el peticionario o servidor de aplicaciones que ha detectado el error.

CONNECT (sin ningún operando), **IN SHARE/EXCLUSIVE MODE**, **USER**, y **USING**

Si existe una conexión, el Tipo 2 se comporta como un Tipo 1. El ID de autorización y el seudónimo de base de datos se colocan en el campo SQLERRMC de la SQLCA. Si no existe una conexión, no se realiza ningún intento de establecer una conexión implícita y los campos SQLERRP y SQLERRMC devuelven un blanco. (Las aplicaciones pueden comprobarse si existe una conexión actual comprobando estos campos.)

CONNECT sin operandos que incluya USER y USING todavía puede conectar un proceso de aplicación a una base de datos mediante la variable de entorno DB2DBDFT. Este método es equivalente a CONNECT RESET de Tipo 2, pero permite el uso de un ID de usuario y una contraseña.

RESET

Equivale a una conexión explícita con la base de datos por omisión, si está disponible. Si no hay una base de datos por omisión que esté disponible, el estado de conexión del proceso de aplicación y los estados de sus conexiones permanecen invariables.

La disponibilidad de una base de datos por omisión se determina de acuerdo con las opciones de instalación, las variables de entorno y los valores de autenticación. Consulte el manual *Guía rápida de iniciación* para obtener información sobre cómo establecer la conexión implícita en la instalación y las variables de entorno, y el manual *Administration Guide* para conseguir información sobre los valores de autenticación.

CONNECT (Tipo 2)

Normas

- Como se describe en el apartado “Opciones que controlan la semántica de la unidad de trabajo distribuida” en la página 54, un conjunto de opciones de conexión rige la semántica de la gestión de conexión. Se asignan valores por omisión a todos los archivos fuente preprocesados. Una aplicación puede constar de múltiples archivos fuente precompilados con diferentes opciones de conexión.

A menos que antes se haya ejecutado un mandato SET CLIENT o una API, las opciones de conexión utilizadas al preprocesar el archivo fuente que contiene la primera sentencia de SQL ejecutada en tiempo de ejecución se convierten en opciones de conexión efectivas.

Si una sentencia CONNECT de un archivo fuente preprocesado con opciones de conexión diferentes se ejecuta posteriormente sin que intervenga en la ejecución ningún mandato SET CLIENT ni API, se produce un error (SQLSTATE 08001). Observe que, una vez que se ha ejecutado un mandato SET CLIENT o una API, se pasan por alto las opciones de conexión utilizadas al preprocesar todos los archivos fuente de la aplicación.

El ejemplo 1 de la página 667 ilustra estas reglas.

- Aunque la sentencia CONNECT TO puede utilizarse para establecer o conmutar conexiones, sólo se aceptará CONNECT TO con la cláusula USER/USING cuando no haya ninguna conexión actual o inactiva con el servidor nombrado. La conexión debe liberarse antes de emitir una conexión con el mismo servidor con la cláusula USER/USING; de lo contrario, se rechazará (SQLSTATE 51022). Libere la conexión emitiendo una sentencia DISCONNECT o una sentencia RELEASE seguida de una sentencia COMMIT.

Notas

- Se da soporte a la conexión implícita para la primera sentencia de SQL en una aplicación con conexiones de Tipo 2. Para ejecutar sentencias de SQL en la base de datos por omisión, primero debe utilizarse la sentencia CONNECT RESET o CONNECT USER/USING para establecer la conexión. La sentencia CONNECT sin operandos visualizará información acerca de la conexión actual si la hay, pero no establecerá una conexión con la base de datos por omisión si no hay ninguna conexión actual.

Comparación de las sentencias CONNECT de Tipo 1 y de Tipo 2:

La semántica de la sentencia CONNECT se determina mediante la opción de precompilador CONNECT o la API SET CLIENT (consulte el apartado “Opciones que controlan la semántica de la unidad de trabajo distribuida” en la página 54). Puede especificarse CONNECT Tipo 1 o CONNECT Tipo 2, y

las sentencias CONNECT en esos programas se conocen, respectivamente, como sentencias CONNECT de Tipo 1 y Tipo 2. Su semántica se describe a continuación:

Utilización de **CONNECT TO**:

Tipo 1	Tipo 2
Cada unidad de trabajo sólo puede establecer conexión con un servidor de aplicaciones.	Cada unidad de trabajo puede establecer conexión con múltiples servidores de aplicaciones.
Se ha de confirmar o retrotraer la unidad de trabajo actual antes de permitir una conexión con otro servidor de aplicaciones.	No es necesario confirmar ni retrotraer la unidad de trabajo actual antes de conectar con otro servidor de aplicaciones.
La sentencia CONNECT establece la conexión actual. Las peticiones SQL posteriores se reenvían a esta conexión hasta que otra CONNECT la modifique.	Igual que CONNECT Tipo 1 si se establece la primera conexión. Si se conmuta a una conexión inactiva y SQLRULES está establecido en STD, debe utilizarse la sentencia SET CONNECTION en su lugar.
Conectar con la conexión actual es válido y no modifica la conexión actual.	Igual que CONNECT Tipo 1 si la opción de precompilador SQLRULES está establecida en DB2. Si SQLRULES está establecida en STD, debe utilizarse la sentencia SET CONNECTION en su lugar.
Conectar con otro servidor de aplicaciones desconecta la conexión actual. La nueva conexión se convierte en la conexión actual. En una unidad de trabajo sólo se mantiene una conexión.	<p>Conectar con otro servidor de aplicaciones pone la conexión actual en el <i>estado inactivo</i>. La nueva conexión se convierte en la conexión actual. Pueden mantenerse múltiples conexiones en una unidad de trabajo.</p> <p>Si CONNECT se ejecuta para un servidor de aplicaciones que está en una conexión inactiva, ésta se convierte en la conexión actual.</p> <p>Conectar con una conexión inactiva mediante CONNECT sólo está permitido si se ha especificado SQLRULES(DB2). Si se ha especificado SQLRULES(STD), debe utilizarse la sentencia SET CONNECTION en su lugar.</p>
No se da soporte a la sentencia SET CONNECTION para conexiones de Tipo 1, pero el único destino válido es la conexión actual.	Se da soporte a la sentencia SET CONNECTION en conexiones de Tipo 2 para cambiar el estado de una conexión inactiva a actual.

CONNECT (Tipo 2)

Utilización de **CONNECT...USER...USING**:

Tipo 1

Conectar con la cláusula **USER...USING** desconecta la conexión actual y establece una nueva conexión con el nombre de autorización y la contraseña proporcionados.

Tipo 2

Conectar con la cláusula **USER/USING** sólo se aceptará cuando no haya una conexión actual o inactiva con el mismo servidor indicado.

Utilización de **CONNECT**, **CONNECT RESET implícitas** y **Desconexión**:

Tipo 1

CONNECT RESET puede utilizarse para desconectar la conexión actual.

Tipo 2

CONNECT RESET equivale a conectar explícitamente con el servidor de aplicaciones por omisión si hay uno definido en el sistema.

La aplicación puede desconectar las conexiones en una **COMMIT** satisfactoria. Antes de la confirmación, utilice la sentencia **RELEASE** para marcar una conexión como pendiente de liberación. Tales conexiones se desconectarán en la siguiente **COMMIT**.

Una alternativa consiste en utilizar las opciones de precompilador **DISCONNECT(EXPLICIT)**, **DISCONNECT(CONDITIONAL)**, **DISCONNECT(AUTOMATIC)** o la sentencia **DISCONNECT** en lugar de la sentencia **RELEASE**.

Tras utilizar **CONNECT RESET** para desconectar la conexión actual, si la siguiente sentencia de **SQL** no es una sentencia **CONNECT**, realizará una conexión implícita con el servidor de aplicaciones por omisión si hay uno definido en el sistema.

CONNECT RESET equivale a una conexión explícita con el servidor de aplicaciones por omisión si hay uno definido en el sistema.

Es un error emitir sentencias **CONNECT RESET** consecutivas.

SÓLO es un error emitir sentencias **CONNECT RESET** consecutivas si se ha especificado **SQLRULES(STD)**, porque esta opción inhabilita el uso de **CONNECT** para la conexión existente.

CONNECT RESET también confirma implícitamente la unidad de trabajo actual.

CONNECT RESET no confirma la unidad de trabajo actual.

Tipo 1	Tipo 2
Si el sistema desconecta una conexión existente por cualquier motivo, las sentencias de SQL posteriores que no sean CONNECT para esta base de datos recibirán un SQLSTATE de 08003.	Si el sistema desconecta una conexión existente, siguen estando permitidas las sentencias COMMIT, ROLLBACK y SET CONNECTION.
La unidad de trabajo se confirmará implícitamente cuando el proceso de aplicación termine de manera satisfactoria.	Igual que el Tipo 1.
Todas las conexiones (sólo una) se desconectan cuando el proceso de aplicación termina.	Todas las conexiones (actuales, inactivas y las marcadas como pendientes de liberación) se desconectan cuando el proceso de aplicación termina.

Anomalías de CONNECT:

Tipo 1	Tipo 2
Sin tener en cuenta si hay una conexión actual cuando CONNECT falla (con un error que no sea que el nombre-servidor no está definido en el directorio local), el proceso de aplicación se coloca en estado no conectado. Las sentencias posteriores que no sean CONNECT recibirán un SQLSTATE de 08003.	Si hay una conexión actual cuando CONNECT falla, la conexión actual no se verá afectada. Si no hay una conexión actual cuando CONNECT falla, el programa pasa a estar en estado no conectado. Las sentencias posteriores que no sean CONNECT recibirán un SQLSTATE de 08003.

Ejemplos

Ejemplo 1: Este ejemplo ilustra la utilización de múltiples programas fuente (mostrados en los recuadros), algunos preprocesados con opciones de conexión distintas (mostradas encima del código) y uno de los cuales contiene una llamada a la API SET CLIENT.

```
PGM1: CONNECT(2) SQLRULES(DB2) DISCONNECT(CONDITIONAL)
```

```
...
exec sql CONNECT TO OTTAWA;
exec sql SELECT col1 INTO :hv1
FROM tbl1;
...
```

```
PGM2: CONNECT(2) SQLRULES(STD) DISCONNECT(AUTOMATIC)
```

```
...
exec sql CONNECT TO QUEBEC;
exec sql SELECT col1 INTO :hv1
FROM tbl2;
...
```

CONNECT (Tipo 2)

PGM3: CONNECT(2) SQLRULES(STD) DISCONNECT(EXPLICIT)

```
...  
SET CLIENT CONNECT 2 SQLRULES DB2 DISCONNECT EXPLICIT 1  
exec sql CONNECT TO LONDON;  
exec sql SELECT col1 INTO  
:hv1 FROM tb13;  
...
```

1 Nota: no es la sintaxis real de la API SET CLIENT

PGM4: CONNECT(2) SQLRULES(DB2) DISCONNECT(CONDITIONAL)

```
...  
exec sql CONNECT TO REGINA;  
exec sql SELECT col1 INTO  
:hv1 FROM tb14;  
...
```

Si la aplicación ejecuta PGM1 y luego PGM2:

- la conexión con OTTAWA ejecuta: connect=2, sqlrules=DB2, disconnect=CONDITIONAL
- la conexión con QUEBEC es anómala con SQLSTATE 08001 porque SQLRULES y DISCONNECT son diferentes.

Si la aplicación ejecuta PGM1 y luego PGM3:

- la conexión con OTTAWA ejecuta: connect=2, sqlrules=DB2, disconnect=CONDITIONAL
- la conexión con LONDON ejecuta: connect=2, sqlrules=DB2, disconnect=EXPLICIT

Esto es correcto porque la API SET CLIENT se ejecuta antes que la segunda sentencia CONNECT.

Si la aplicación ejecuta PGM1 y luego PGM4:

- la conexión con OTTAWA ejecuta: connect=2, sqlrules=DB2, disconnect=CONDITIONAL
- la conexión con REGINA ejecuta: connect=2, sqlrules=DB2, disconnect=CONDITIONAL

Esto es correcto porque las opciones de preprocesador para PGM1 son las mismas que para PGM4.

Ejemplo 2:

Este ejemplo muestra las interrelaciones entre las sentencias CONNECT (Tipo 2), SET CONNECTION, RELEASE y DISCONNECT. S0, S1, S2 y S3 representan cuatro servidores.

CONNECT (Tipo 2)

Secuencia	Sentencia	Servidor actual	Conexiones inactivas	Pendiente de liberación
0	Ninguna sentencia	Ninguno	Ninguno	Ninguno
1.	SELECT * FROM TBLA	S0 (valor por omisión)	Ninguno	Ninguno
2	CONNECT TO S1 SELECT * FROM TBLB	S1 S1	S0 S0	Ninguno Ninguno
3	CONNECT TO S2 UPDATE TBLC SET ...	S2 S2	S0, S1 S0, S1	Ninguno Ninguno
4	CONNECT TO S3 SELECT * FROM TBLD	S3 S3	S0, S1, S2 S0, S1, S2	Ninguno Ninguno
5	SET CONNECTION S2	S2	S0, S1, S3	Ninguno
6	RELEASE S3	S2	S0, S1	S3
7	COMMIT	S2	S0, S1	Ninguno
8	SELECT * FROM TBLE	S2	S0, S1	Ninguno
9	DISCONNECT S1 SELECT * FROM TBLF	S2 S2	S0 S0	Ninguno Ninguno

CREATE ALIAS

CREATE ALIAS

La sentencia CREATE ALIAS define un seudónimo para una tabla, una vista, un apodo u otro seudónimo.

Invocación

Esta sentencia se puede intercalar en un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámicas. Es una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

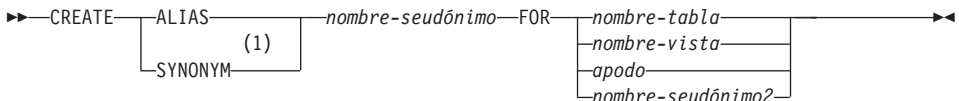
Autorización

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Autorización IMPLICIT_SCHEMA para la base de datos, si no existe el nombre de esquema, implícito o explícito, del seudónimo
- Privilegio CREATEIN para el esquema, si el nombre de esquema del seudónimo hace referencia a un esquema existente.

Para utilizar el objeto referenciado a través del seudónimo, son necesarios los mismos privilegios para el objeto que los que serían necesarios si se utilizara el propio objeto.

Sintaxis



Notas:

- 1 Se acepta CREATE SYNONYM como una alternativa para CREATE ALIAS para la tolerancia de sintaxis de las sentencias CREATE SYNONYM existentes de otras implantaciones de SQL.

Descripción

nombre-seudónimo

Indica el nombre del seudónimo. El nombre no debe identificar una tabla, una vista, un apodo o un seudónimo que exista en la base de datos actual.

Si se especifica un nombre de dos partes, el nombre de esquema no puede empezar por "SYS" (SQLSTATE 42939).

Las reglas que se emplean para definir un seudónimo son las mismas que las que se emplean para definir un nombre de tabla.

FOR *nombre-tabla, nombre-vista, apodo, o nombre-seudónimo2*

Identifica la tabla, la vista, el apodo o el seudónimo para el que se ha definido *nombre-seudónimo*. En caso de proporcionar otro seudónimo (*nombre-seudónimo2*), éste no debe ser el mismo que el nuevo *nombre-seudónimo* que se está definiendo (en su forma completamente calificada). El *nombre-tabla* no puede ser una tabla temporal declarada (SQLSTATE 42995).

Notas

- La definición del seudónimo recién creado se almacena en SYSCAT.TABLES.
- Se puede definir un seudónimo para un objeto que no exista en el momento de la definición. Si no existe, aparece un mensaje de aviso (SQLSTATE 01522). No obstante, el objeto al que se hace referencia debe existir al compilar una sentencia de SQL que contenga dicho seudónimo; de lo contrario, aparece un mensaje de error (SQLSTATE 52004).
- Se puede definir un seudónimo para hacer referencia a otro seudónimo como parte de una cadena de seudónimos, pero dicha cadena está sujeta a las mismas restricciones que un seudónimo normal cuando se utiliza en una sentencia de SQL. La cadena de seudónimos se resuelve de la misma manera que un solo seudónimo. Si un seudónimo que se utiliza en una definición de vista, en una sentencia de un paquete o en un desencadenante apunta a una cadena de seudónimos, esta relación de dependencia queda registrada para la vista, paquete o desencadenante en cada uno de los seudónimos de la cadena. Los ciclos repetidos no están permitidos en una cadena de seudónimos y se detectan durante el proceso de definición.
- La creación de un seudónimo con un nombre de esquema que todavía no existe dará como resultado la creación implícita de dicho esquema siempre que el ID de autorización de la sentencia tenga autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN en el esquema se otorga a PUBLIC.

Ejemplos

Ejemplo 1: HEDGES intenta crear un seudónimo para una tabla T1 (ambos sin calificar).

```
CREATE ALIAS A1 FOR T1
```

Se crea el seudónimo HEDGES.A1 para HEDGES.T1.

Ejemplo 2: HEDGES intenta crear un seudónimo para una tabla (ambos calificados).

```
CREATE ALIAS HEDGES.A1 FOR MCKNIGHT.T1
```

Se crea el seudónimo HEDGES.A1 para MCKNIGHT.T1.

CREATE ALIAS

Ejemplo 3: HEDGES intenta crear un seudónimo para una tabla (el seudónimo en un esquema diferente; HEDGES no es DBADM; HEDGES no tiene el privilegio CREATEIN en el esquema MCKNIGHT).

```
CREATE ALIAS MCKNIGHT.A1 FOR MCKNIGHT.T1
```

Este ejemplo falla (SQLSTATE 42501).

Ejemplo 4: HEDGES intenta crear un seudónimo para una tabla no definida (ambos calificados; FUZZY.WUZZY no existe).

```
CREATE ALIAS HEDGES.A1 FOR FUZZY.WUZZY
```

Esta sentencia resulta satisfactoria pero emite un mensaje de aviso (SQLSTATE 01522).

Ejemplo 5: HEDGES intenta crear un seudónimo para un seudónimo (ambos calificados).

```
CREATE ALIAS HEDGES.A1 FOR MCKNIGHT.T1  
CREATE ALIAS HEDGES.A2 FOR HEDGES.A1
```

La primera sentencia resulta satisfactoria (como en el ejemplo 2).

La segunda sentencia es satisfactoria y crea una cadena de seudónimos, según la cual HEDGES.A2 hace referencia a HEDGES.A1, que a su vez hace referencia a MCKNIGHT.T1. Observe que no importa si HEDGES tiene privilegios o no sobre MCKNIGHT.T1. El seudónimo se crea sean cuales sean los privilegios de las tablas.

Ejemplo 6: Designar A1 como seudónimo para el apodo FUZZYBEAR.

```
CREATE ALIAS A1 FOR FUZZYBEAR
```

Ejemplo 7: Una gran organización tiene un departamento financiero con el número D108 y un departamento de personal con el número D577. D108 conserva determinada información en una tabla que reside en DB2 RDBMS. D577 conserva determinados registros en una tabla que reside en Oracle RDBMS. Un DBA define los dos RDBMS como fuentes de datos de un sistema federado y asigna a las tablas los apodos de DEPTD108 y DEPTD577, respectivamente. El usuario de un sistema federado necesita crear vínculos entre estas tablas, pero preferiría hacer referencia a ellas por los nombres que tienen más sentido que sus apodos alfanuméricos. Por lo tanto, el usuario define FINANCE como un seudónimo para DEPTD108 y PERSONNEL como un seudónimo para DEPTD577.

```
CREATE ALIAS FINANCE FOR DEPTD108  
CREATE ALIAS PERSONNEL FOR DEPTD577
```

CREATE BUFFERPOOL

La sentencia CREATE BUFFERPOOL crea una agrupación de almacenamientos intermedios nueva para que la utilice el gestor de bases de datos. Aunque la definición de agrupación de almacenamientos intermedios es transaccional y las entradas se reflejarán en las tablas del catálogo en la confirmación, la agrupación de almacenamientos intermedios no se activará hasta la próxima vez que se inicie la base de datos.

En una base de datos particionada, se especifica una definición de agrupación de almacenamientos intermedios por omisión para cada partición o nodo, con la posibilidad de alterar temporalmente el tamaño en particiones o nodos específicos. También, en una base de datos particionada, la agrupación de almacenamientos intermedios por omisión se define en todas las particiones a menos que se especifiquen grupos de nodos. Si se especifican grupos de nodos, la agrupación de almacenamientos intermedios sólo se crearán en particiones que están en estos grupos de nodos.

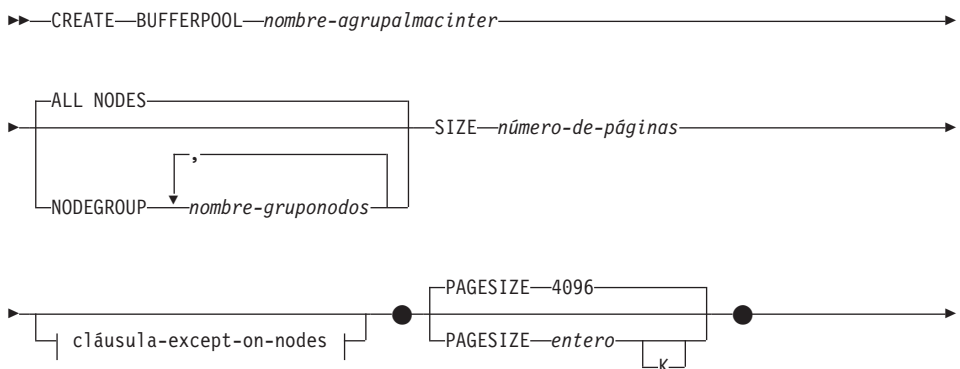
Invocación

Esta sentencia puede incluirse en un programa de aplicación o emitirse de forma interactiva. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

Autorización

El ID de autorización de la sentencia debe tener autorización SYSCTRL o SYSADM.

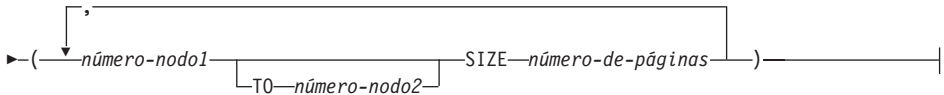
Sintaxis



CREATE BUFFERPOOL



cláusula-except-on-nodes:



Descripción

nombre-agrupalmcinter

Indica el nombre de la agrupación de almacenamientos intermedios. Este nombre sólo se compone de una parte. Se trata de un identificador de SQL (ordinario o bien delimitado). El *nombre-agrupalmcinter* no debe identificar una agrupación de almacenamientos intermedios que ya exista en un catálogo (SQLSTATE 42710). El *nombre-agrupalmcinter* no debe empezar por los caracteres "SYS" o "IBM" (SQLSTATE 42939).

ALL NODES

Esta agrupación de almacenamientos intermedios se creará en todas las particiones de la base de datos.

NODEGROUP *nombre-gruponodos, ...*

Identifica el grupo de nodos o los grupos de nodos a los que se puede aplicar la definición de agrupación de almacenamientos intermedios. Si se especifica esto, esta agrupación de almacenamientos intermedios sólo se creará en las particiones de estos grupos de nodos. Cada grupo de nodos debe existir actualmente en la base de datos (SQLSTATE 42704). Si no se especifica la palabra clave NODEGROUP, entonces esta agrupación de almacenamientos intermedios se creará en todas las particiones (y en cualquier partición que se añada con posterioridad a la base de datos).

SIZE *número-de-páginas*

El tamaño de agrupación de almacenamientos intermedios especificado como el número de páginas.⁷⁰ En una base de datos particionada, será el tamaño por omisión para todas las particiones en las que exista la agrupación de almacenamientos intermedios.

⁷⁰ El tamaño puede especificarse con un valor de (-1), que indica que el tamaño de la agrupación de almacenamientos intermedios debe obtenerse del parámetro de configuración BUFFPAGE de la base de datos.

cláusula-except-on-nodes

Especifica la partición o particiones para las que el tamaño de la agrupación de almacenamientos intermedios será diferente del valor por omisión. Si no se especifica esta cláusula, entonces todas las particiones tendrán el mismo tamaño que el especificado para esta agrupación de almacenamientos intermedios.

EXCEPT ON NODES

Palabras clave que indican que se han especificado particiones específicas. NODE es un sinónimo de NODES.

número-nodo1

Especifica un número de partición específica que se incluye en las particiones para las que se crea la agrupación de almacenamientos intermedios.

TO *número-nodo2*

Especifique un rango de números de partición. El valor de *número-nodo2* debe ser mayor o igual que el valor de *número-nodo1* (SQLSTATE 428A9). Todas las particiones entre los números de partición, inclusive los especificados, deben incluirse en las particiones para las que se ha creado la agrupación de almacenamientos intermedios (SQLSTATE 42729).

SIZE *número-de-páginas*

El tamaño de agrupación de almacenamientos intermedios especificado como el número de páginas.

PAGESIZE *entero* [K]

Define el tamaño de las páginas utilizadas para la agrupación de almacenamientos intermedios. Los valores válidos de *entero* sin el sufijo K son 4 096, 8 192, 16 384 ó 32 768. Los valores válidos de *entero* con el sufijo K son 4, 8, 16 ó 32. Se produce un error si el tamaño de página no es uno de estos valores (SQLSTATE 428DE). El valor por omisión es páginas de 4 096 bytes (4K). Se permite cualquier número de espacios entre *entero* y K, incluso ningún espacio.

EXTENDED STORAGE

Si se activa la configuración del almacenamiento ampliado,⁷¹ las páginas que se están migrando fuera de esta agrupación de almacenamientos intermedios se pondrán en antememoria en el almacenamiento ampliado.

NOT EXTENDED STORAGE

Incluso si se activa la configuración del almacenamiento ampliado de la

⁷¹ Se activa la configuración del almacenamiento ampliado si se establecen los parámetros de configuración de base de datos NUM_ESTORE_SEGS y ESTORE_SEG_SIZE en valores que no sean cero. Vea el manual *Administration Guide* para conocer detalles.

CREATE BUFFERPOOL

base de datos, las páginas que migran fuera de esta agrupación de almacenamientos intermedios, NO se pondrán en antememoria en el almacenamiento ampliado.

Notas

- Hasta la próxima vez que se inicie la base de datos, cualquier espacio de tablas que se cree utilizará una agrupación de almacenamientos intermedios ya activa con el mismo tamaño de página. Se ha de reiniciar la base de datos para que la asignación del espacio de tablas a la nueva agrupación de almacenamientos intermedios tenga efecto.
- Debe haber suficiente memoria real en la máquina para el total de las agrupaciones de almacenamientos intermedios, así como para el resto de necesidades de espacio del gestor de bases de datos y de las aplicaciones. Si DB2 no puede obtener la memoria total para todas las agrupaciones de almacenamientos intermedios, intentará iniciar sólo la agrupación de almacenamientos intermedios por omisión. Si no es satisfactorio, iniciará una agrupación de almacenamientos intermedios por omisión mínima. En cualquiera de estos casos, se devolverá un aviso al usuario (SQLSTATE 01626) y las páginas para todos los espacios de tablas utilizarán la agrupación de almacenamientos intermedios por omisión.

CREATE DISTINCT TYPE

La sentencia CREATE DISTINCT TYPE define un tipo diferenciado. El tipo diferenciado siempre tiene su fuente en los tipos de datos incorporados. La ejecución satisfactoria de la sentencia también genera funciones para convertir entre el tipo diferenciado y su tipo fuente y, opcionalmente, genera el soporte para utilizar los operadores de comparación (=, <>, <, <=, > y >=) con el tipo diferenciado.

Invocación

Esta sentencia se puede intercalar en un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámicas. Es una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

Autorización

El ID de autorización de la sentencia debe tener como mínimo uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema del tipo diferenciado no hace referencia a un esquema existente.
- Privilegio CREATEIN para el esquema, si el nombre de esquema del tipo diferenciado hace referencia a un esquema existente.

Sintaxis

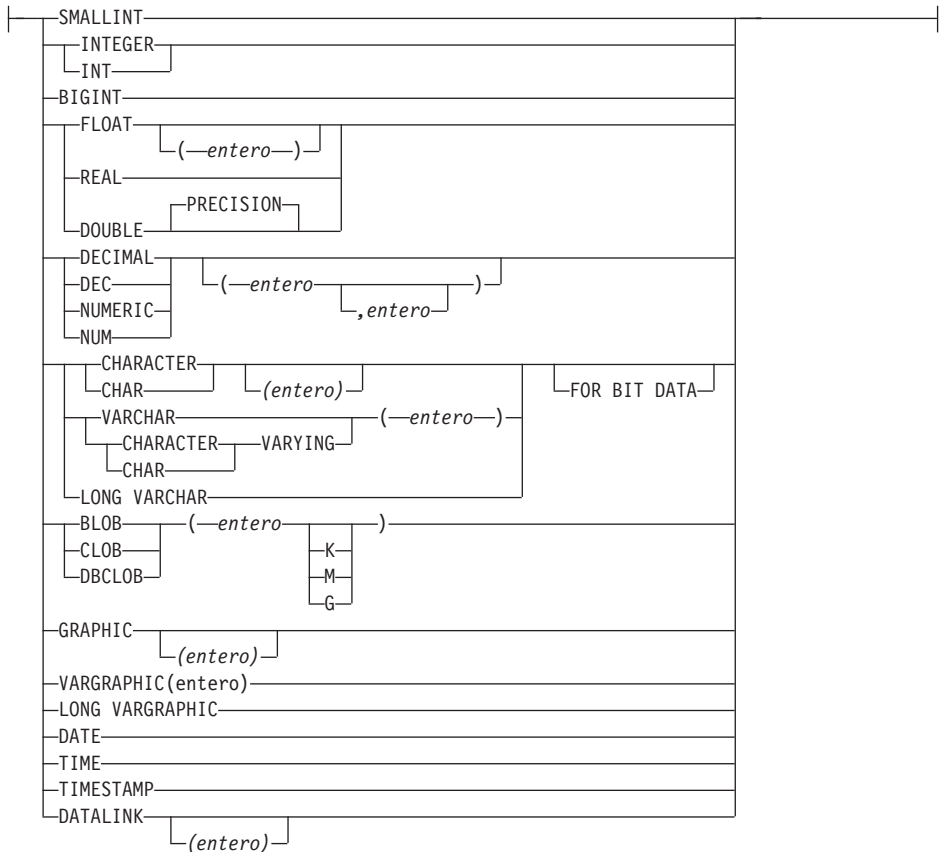
```

▶▶—CREATE DISTINCT TYPE—nombre-tipo-diferenciado—AS—▶▶
▶ | tipo-datos-fuente | —WITH COMPARISONS—▶▶(1)

```

tipo-datos-fuente:

CREATE DISTINCT TYPE



Notas:

- 1 Necesario para todos los tipos-datos-fuente excepto LOB, LONG VARCHAR, LONG VARGRAPHIC y DATALINK, a los que no se da soporte.

Descripción

nombre-tipo-diferenciado

Indica el nombre del tipo diferenciado. El nombre, incluido el calificador implícito o explícito, no debe designar un tipo diferenciado descrito en el catálogo. El nombre no calificado no debe ser igual que el nombre de un tipo de datos fuente ni BOOLEAN (SQLSTATE 42918).

En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador

para los nombres de objetos no calificados. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL.

El nombre de esquema (implícito o explícito) no debe ser mayor que 8 bytes (SQLSTATE 42622).

Algunos nombres utilizados como palabras clave en predicados están reservados para que los utilice el sistema y no pueden utilizarse como *nombre-tipo-diferenciado*. Estos nombres son SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH y los operadores de comparación, tal como se describe en “Predicado básico” en la página 213. Cualquier incumplimiento de esta regla provocará la aparición de un error (SQLSTATE 42939).

Si se especifica un *nombre-tipo-diferenciado*, el nombre de esquema no puede empezar por "SYS"; de lo contrario, se genera un error (SQLSTATE 42939).

tipo-datos-fuente

Especifica el tipo de datos utilizado como base para la representación interna del tipo diferenciado. Para obtener información acerca de la asociación de tipos diferenciados con otros tipos de datos, consulte el apartado “Tipos diferenciados” en la página 96. Para obtener información acerca de los tipos de datos, consulte el apartado “CREATE TABLE” en la página 835.

WITH COMPARISONS

Especifica que se han de crear los operadores de comparación generados por el sistema para comparar dos instancias de un tipo diferenciado. Estas palabras clave no deben especificarse si el tipo-datos-fuente es BLOB, CLOB, DBCLOB, LONG VARCHAR, LONG VARGRAPHIC o DATALINK; de lo contrario, se devolverá un aviso (SQLSTATE 01596) y no se generarán los operadores de comparación. Para todos los demás tipos-datos-fuente, son necesarias las palabras clave WITH COMPARISONS.

Notas

- La creación de un tipo diferenciado con un nombre de esquema que todavía no existe dará como resultado la creación implícita de ese esquema siempre que el ID de autorización de la sentencia tenga la autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN en el esquema se otorga a PUBLIC.
- Se generan las funciones siguientes para convertir el tipo de datos utilizando el tipo fuente como origen o destino de la conversión:
 - Una función para convertir del tipo diferenciado al tipo fuente
 - Una función para convertir del tipo fuente al tipo diferenciado
 - Una función para convertir de INTEGER al tipo diferenciado si el tipo fuente es SMALLINT

CREATE DISTINCT TYPE

- Una función para convertir de VARCHAR al tipo diferenciado si el tipo fuente es CHAR
- Una función para convertir de VARGRAPHIC al tipo diferenciado si el tipo fuente es GRAPHIC.

En general estas funciones tendrán el siguiente formato:

```
CREATE FUNCTION nombre-tipo-fuente (nombre-tipo-diferenciado)  
RETURNS nombre-tipo-fuente ...
```

```
CREATE FUNCTION nombre-tipo-diferenciado (nombre-tipo-fuente)  
RETURNS nombre-tipo-diferenciado ...
```

En los casos en que el tipo fuente es un tipo con parámetros, la función para convertir el tipo diferenciado al tipo fuente tendrá como nombre de función el nombre del tipo fuente sin los parámetros (consulte los detalles en la Tabla 22 en la página 681). El tipo de valor de retorno de esta función incluirá los parámetros dados en la sentencia CREATE DISTINCT TYPE. La función para convertir del tipo fuente al tipo diferenciado tendrá un parámetro de entrada cuyo tipo es el tipo fuente incluyendo sus parámetros. Por ejemplo,

```
CREATE DISTINCT TYPE T_SHOESIZE AS CHAR(2)  
WITH COMPARISONS
```

```
CREATE DISTINCT TYPE T_MILES AS DOUBLE  
WITH COMPARISONS
```

generará las funciones siguientes:

```
FUNCTION CHAR (T_SHOESIZE) RETURNS CHAR (2)
```

```
FUNCTION T_SHOESIZE (CHAR (2))  
RETURNS T_SHOESIZE
```

```
FUNCTION DOUBLE (T_MILES) RETURNS DOUBLE
```

```
FUNCTION T_MILES (DOUBLE) RETURNS T_MILES
```

El esquema de las funciones de conversión generadas es el mismo que el esquema del tipo diferenciado. No debe existir ninguna otra función con este nombre y con la misma signatura en la base de datos (SQLSTATE 42710).

La tabla siguiente ofrece los nombres de las funciones para convertir del tipo diferenciado al tipo fuente y del tipo fuente al tipo diferenciado para todos los tipos de datos predefinidos.

Tabla 22. Funciones CAST en tipos diferenciados

Nombre de tipo fuente	Nombre de función	Parámetro	Tipo de retorno
CHAR	<i>nombre-tipo-diferenciado</i>	CHAR (n)	<i>nombre-tipo-diferenciado</i>
	CHAR	<i>nombre-tipo-diferenciado</i>	CHAR (n)
VARCHAR	<i>nombre-tipo-diferenciado</i>	VARCHAR (n)	<i>nombre-tipo-diferenciado</i>
	VARCHAR	<i>nombre-tipo-diferenciado</i>	VARCHAR (n)
LONG VARCHAR	<i>nombre-tipo-diferenciado</i>	LONG VARCHAR	<i>nombre-tipo-diferenciado</i>
	LONG_VARCHAR	<i>nombre-tipo-diferenciado</i>	LONG VARCHAR
CLOB	<i>nombre-tipo-diferenciado</i>	CLOB (n)	<i>nombre-tipo-diferenciado</i>
	CLOB	<i>nombre-tipo-diferenciado</i>	CLOB (n)
BLOB	<i>nombre-tipo-diferenciado</i>	BLOB (n)	<i>nombre-tipo-diferenciado</i>
	BLOB	<i>nombre-tipo-diferenciado</i>	BLOB (n)
GRAPHIC	<i>nombre-tipo-diferenciado</i>	GRAPHIC (n)	<i>nombre-tipo-diferenciado</i>
	GRAPHIC	<i>nombre-tipo-diferenciado</i>	GRAPHIC (n)
	<i>nombre-tipo-diferenciado</i>	VARGRAPHIC (n)	<i>nombre-tipo-diferenciado</i>
VARGRAPHIC	<i>nombre-tipo-diferenciado</i>	VARGRAPHIC (n)	<i>nombre-tipo-diferenciado</i>
	VARGRAPHIC	<i>nombre-tipo-diferenciado</i>	VARGRAPHIC (n)
LONG VARGRAPHIC	<i>nombre-tipo-diferenciado</i>	LONG VARGRAPHIC	<i>nombre-tipo-diferenciado</i>
	LONG_VARGRAPHIC	<i>nombre-tipo-diferenciado</i>	LONG VARGRAPHIC
DBCLOB	<i>nombre-tipo-diferenciado</i>	DBCLOB (n)	<i>nombre-tipo-diferenciado</i>
	DBCLOB	<i>nombre-tipo-diferenciado</i>	DBCLOB (n)
SMALLINT	<i>nombre-tipo-diferenciado</i>	SMALLINT	<i>nombre-tipo-diferenciado</i>
	<i>nombre-tipo-diferenciado</i>	INTEGER	<i>nombre-tipo-diferenciado</i>
	SMALLINT	<i>nombre-tipo-diferenciado</i>	SMALLINT
INTEGER	<i>nombre-tipo-diferenciado</i>	INTEGER	<i>nombre-tipo-diferenciado</i>
	INTEGER	<i>nombre-tipo-diferenciado</i>	INTEGER
BIGINT	<i>nombre-tipo-diferenciado</i>	BIGINT	<i>nombre-tipo-diferenciado</i>
	BIGINT	<i>nombre-tipo-diferenciado</i>	BIGINT
DECIMAL	<i>nombre-tipo-diferenciado</i>	DECIMAL (p,s)	<i>nombre-tipo-diferenciado</i>
	DECIMAL	<i>nombre-tipo-diferenciado</i>	DECIMAL (p,s)
NUMERIC	<i>nombre-tipo-diferenciado</i>	DECIMAL (p,s)	<i>nombre-tipo-diferenciado</i>
	DECIMAL	<i>nombre-tipo-diferenciado</i>	DECIMAL (p,s)

CREATE DISTINCT TYPE

Tabla 22. Funciones CAST en tipos diferenciados (continuación)

Nombre de tipo fuente	Nombre de función	Parámetro	Tipo de retorno
REAL	<i>nombre-tipo-diferenciado</i>	REAL	<i>nombre-tipo-diferenciado</i>
	<i>nombre-tipo-diferenciado</i>	DOUBLE	<i>nombre-tipo-diferenciado</i>
	REAL	<i>nombre-tipo-diferenciado</i>	REAL
FLOAT(<i>n</i>) donde $n \leq 24$	<i>nombre-tipo-diferenciado</i>	REAL	<i>nombre-tipo-diferenciado</i>
	<i>nombre-tipo-diferenciado</i>	DOUBLE	<i>nombre-tipo-diferenciado</i>
	REAL	<i>nombre-tipo-diferenciado</i>	REAL
FLOAT(<i>n</i>) donde $n > 24$	<i>nombre-tipo-diferenciado</i>	DOUBLE	<i>nombre-tipo-diferenciado</i>
	DOUBLE	<i>nombre-tipo-diferenciado</i>	DOUBLE
FLOAT	<i>nombre-tipo-diferenciado</i>	DOUBLE	<i>nombre-tipo-diferenciado</i>
	DOUBLE	<i>nombre-tipo-diferenciado</i>	DOUBLE
DOUBLE	<i>nombre-tipo-diferenciado</i>	DOUBLE	<i>nombre-tipo-diferenciado</i>
	DOUBLE	<i>nombre-tipo-diferenciado</i>	DOUBLE
DOUBLE PRECISION	<i>nombre-tipo-diferenciado</i>	DOUBLE	<i>nombre-tipo-diferenciado</i>
	DOUBLE	<i>nombre-tipo-diferenciado</i>	DOUBLE
DATE	<i>nombre-tipo-diferenciado</i>	DATE	<i>nombre-tipo-diferenciado</i>
	DATE	<i>nombre-tipo-diferenciado</i>	DATE
TIME	<i>nombre-tipo-diferenciado</i>	TIME	<i>nombre-tipo-diferenciado</i>
	TIME	<i>nombre-tipo-diferenciado</i>	TIME
TIMESTAMP	<i>nombre-tipo-diferenciado</i>	TIMESTAMP	<i>nombre-tipo-diferenciado</i>
	TIMESTAMP	<i>nombre-tipo-diferenciado</i>	TIMESTAMP
DATALINK	<i>nombre-tipo-diferenciado</i>	DATALINK	<i>nombre-tipo-diferenciado</i>
	DATALINK	<i>nombre-tipo-diferenciado</i>	DATALINK

Nota: NUMERIC y FLOAT no son aconsejables para crear un tipo de datos definido por el usuario para una aplicación portable. Deben utilizarse DECIMAL y DOUBLE en su lugar.

Las funciones descritas en la tabla anterior son las únicas funciones que se generan automáticamente cuando se definen los tipos diferenciados. En consecuencia, ninguna de las funciones incorporadas (AVG, MAX, LENGTH, etc.) están soportadas en los tipos diferenciados hasta que se utiliza la sentencia CREATE FUNCTION (consulte el apartado “CREATE FUNCTION” en la página 695) para registrar las funciones definidas por el usuario para el tipo diferenciado, donde estas funciones definidas por el usuario tienen su fuente en las funciones incorporadas adecuadas. En particular, observe que es posible registrar funciones definidas por el usuario que tienen su fuente en funciones de columna incorporadas.

Cuando se crea un tipo diferenciado utilizando la cláusula `WITH COMPARISONS`, se crean operadores de comparación generados por el sistema. La creación de estos operadores de comparación generará entradas en la vista de catálogo `SYSCAT.FUNCTIONS` para las nuevas funciones.

El nombre de esquema del tipo diferenciado debe incluirse en la vía de acceso de SQL (consulte “`SET PATH`” en la página 1181 o la opción `FUNCPATH BIND` tal como se describe en el manual *Application Development Guide*) para la utilización satisfactoria de estos operadores y funciones de conversión en las sentencias de SQL.

Ejemplos

Ejemplo 1: Cree un tipo diferenciado llamado `SHOESIZE` que se base en un tipo de datos `INTEGER`.

```
CREATE DISTINCT TYPE SHOESIZE AS INTEGER WITH COMPARISONS
```

Esto también dará como resultado la creación de operadores de comparación (`=`, `<>`, `<`, `<=`, `>`, `>=`) y que la función de conversión `INTEGER(SHOESIZE)` devuelva `INTEGER` y la función de conversión `SHOESIZE(INTEGER)` devuelva `SHOESIZE`.

Ejemplo 2: Cree un tipo diferenciado llamado `MILES` que se base en un tipo de datos `DOUBLE`.

```
CREATE DISTINCT TYPE MILES AS DOUBLE WITH COMPARISONS
```

Esto también dará como resultado la creación de operadores de comparación (`=`, `<>`, `<`, `=`, `>`, `>=`) y que la función de conversión `DOUBLE(MILES)` devuelva `DOUBLE` y la función de conversión `MILES(DOUBLE)` devuelva `MILES`.

CREATE EVENT MONITOR

CREATE EVENT MONITOR

La sentencia CREATE EVENT MONITOR define un supervisor que registrará ciertos sucesos que se produzcan cuando se utilice la base de datos. La definición de cada supervisor de sucesos especifica también el lugar donde la base de datos debe registrar los sucesos.

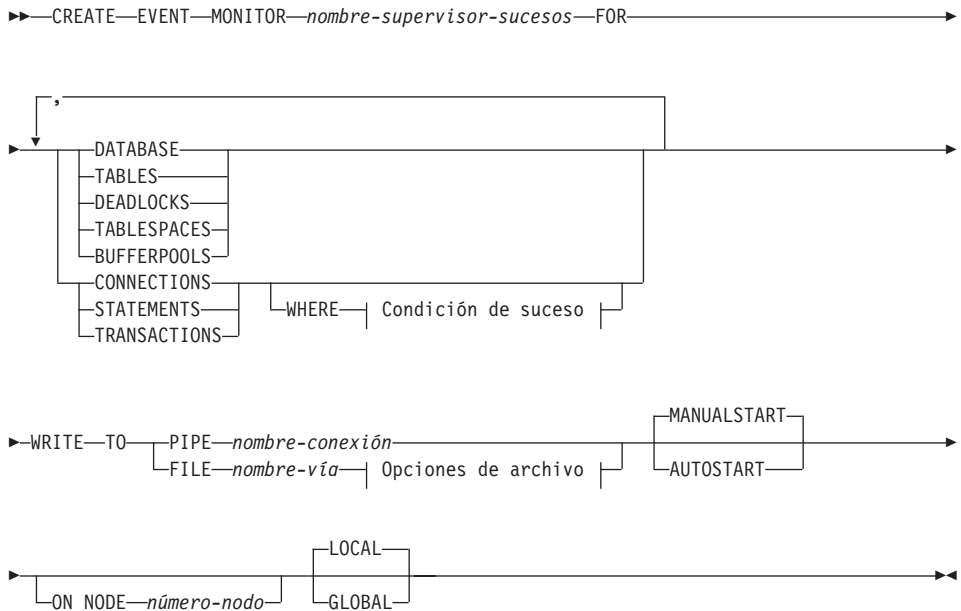
Invocación

Esta sentencia puede incluirse en un programa de aplicación o emitirse de forma interactiva. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

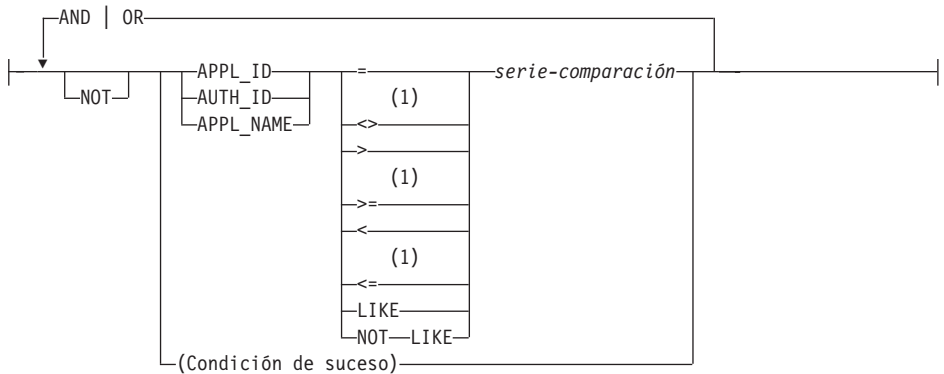
Autorización

Los privilegios del ID de autorización deben incluir la autorización SYSADM o DBADM (SQLSTATE 42502).

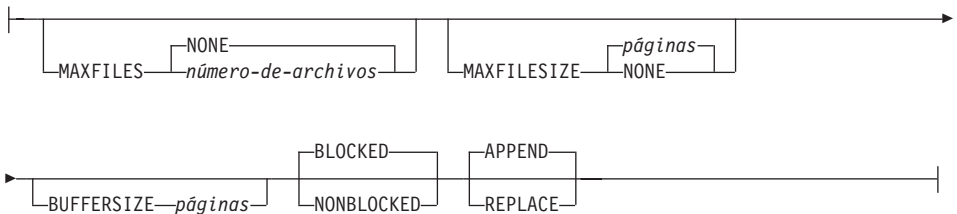
Sintaxis



Condición de suceso:



Opciones de archivo:



Notas:

- 1 También se da soporte a otras formas de estos operadores. Vea "Predicado básico" en la página 213 para conocer más detalles.

Descripción

nombre-supervisor-sucesos

Indica el nombre del supervisor de sucesos. Este nombre sólo se compone de una parte. Se trata de un identificador de SQL (ordinario o bien delimitado). El *nombre-supervisor-sucesos* no debe identificar ningún supervisor de sucesos que ya exista en el catálogo (SQLSTATE 42710).

FOR

Introduce el tipo de suceso que se ha de registrar.

DATABASE

Especifica que el supervisor de sucesos registre un suceso de base de datos cuando se desconecte la última aplicación de la base de datos.

TABLES

Especifica que el supervisor de sucesos registre un suceso de tabla para cada tabla activa cuando se desconecte la última aplicación de la

CREATE EVENT MONITOR

base de datos. Una tabla activa es una tabla que ha cambiado desde la primera conexión con la base de datos.

DEADLOCKS

Especifica que el supervisor de sucesos registre un suceso de punto muerto siempre que se produzca un punto muerto.

TABLESPACES

Especifica que el supervisor de sucesos registre un suceso de espacio de tablas para cada espacio de tablas cuando se desconecte la última aplicación de la base de datos.

BUFFERPOOLS

Especifica que el supervisor de sucesos registre un suceso de agrupación de almacenamientos intermedios cuando se desconecte la última aplicación de la base de datos.

CONNECTIONS

Especifica que el supervisor de sucesos registre un suceso de conexión cuando una aplicación se desconecta de la base de datos.

STATEMENTS

Especifica que el supervisor de sucesos registre un suceso de sentencia siempre que una sentencia de SQL termine su ejecución.

TRANSACTIONS

Especifica que el supervisor de sucesos registre un suceso de transacción siempre que se complete una transacción (es decir, siempre que haya una operación de confirmación o retroacción).

WHERE *condición de suceso*

Define un filtro que determina qué conexiones hacen que se produzca un suceso de CONNECTION, STATEMENT o TRANSACTION. Si el resultado de la condición de suceso es TRUE para una conexión en particular, dicha conexión generará los sucesos pedidos.

Esta cláusula es una forma especial de la cláusula WHERE que no debe confundirse con una condición de búsqueda estándar.

Para determinar si una aplicación generará los sucesos para un supervisor de sucesos en particular, se evalúa la cláusula WHERE:

1. En cada conexión activa cuando se activa un supervisor de sucesos.
2. Posteriormente en cada conexión nueva con la base de datos en el tiempo de conexión.

La cláusula WHERE no se evalúa para cada suceso.

Si no se especifica ninguna cláusula WHERE, se supervisarán todos los sucesos del tipo de suceso especificado.

APPL_ID

Especifica que el ID de aplicación para cada conexión debe compararse con la *serie-comparación* para determinar si la conexión debe generar sucesos de CONNECTION, STATEMENT o TRANSACTION (lo que se haya especificado).

AUTH_ID

Especifica que el ID de autorización de cada conexión debe compararse con la *serie-comparación* para determinar si la conexión debe generar sucesos de CONNECTION, STATEMENT or TRANSACTION (lo que se haya especificado).

APPL_NAME

Especifica que el nombre de programa de aplicación de cada conexión debe compararse con la *serie-comparación* para determinar si la conexión debe generar sucesos de CONNECTION, STATEMENT o TRANSACTION (lo que se haya especificado).

El nombre de programa de aplicación son los 20 primeros bytes del nombre de archivo del programa de aplicación, después del último separador de la vía de acceso.

serie-comparación

La serie que se ha de comparar con el APPL_ID, AUTH_ID o APPL_NAME de cada aplicación que se conecta a la base de datos. *serie-comparación* debe ser una constante de serie (es decir, las variables del lenguaje principal y otras expresiones de serie no están permitidas).

WRITE TO

Introduce el destino para los datos.

PIPE

Especifica que el destino para los datos del supervisor de sucesos es una conexión con nombre. El supervisor de sucesos graba los datos en la conexión en una sola corriente (es decir, como si fuera un solo archivo infinitamente largo). Cuando se graban los datos en una conexión, el supervisor de sucesos no realiza grabaciones por bloques. Si no hay espacio en el almacenamiento intermedio de conexión, el supervisor de sucesos desechará los datos. Es responsabilidad de la aplicación supervisora el leer los datos pronto si desea asegurar que no se pierdan datos.

nombre-conexión

El nombre de la conexión (FIFO en AIX) en la que el supervisor de sucesos grabará los datos.

Las reglas de denominación para las conexiones son específicas de las plataformas. En sistemas operativos UNIX, los nombres de conexiones se tratan como nombres de archivo. Como resultado,

CREATE EVENT MONITOR

están permitidos los nombres de conexiones relativos y se tratan como nombres-vía relativas (consulte *nombre-vía* más abajo). Sin embargo, en OS/2, Windows 95 y Windows NT, hay una sintaxis especial para un nombre de conexión. Como resultado, en OS/2, Windows 95 y Windows NT se necesitan nombres de conexiones absolutos.

La existencia de una conexión no se comprobará en el tiempo de creación del supervisor de sucesos. Es responsabilidad de la aplicación supervisora haber creado y abierto la conexión para lectura en el momento en que se activa el supervisor de sucesos. Si la conexión no está disponible en ese momento, el supervisor de sucesos se desactivará por sí solo y anotará cronológicamente un error. (Es decir, si se ha activado el supervisor de sucesos en el momento del inicio de la base de datos como resultado de la opción AUTOSTART, el supervisor de sucesos anotará un error en el registro cronológico de errores del sistema.) Si se activa el supervisor de sucesos mediante la sentencia SET EVENT MONITOR STATE SQL, dicha sentencia fallará (SQLSTATE 58030).

FILE

Indica que el destino para los datos del supervisor de sucesos es un archivo (o conjunto de archivos). El supervisor de sucesos graba la corriente de datos como una serie de archivos numerados de 8 caracteres, con la extensión "evt". (por ejemplo, 00000000.evt, 00000001.evt y 00000002.evt). Los datos deben considerarse un archivo lógico incluso cuando se dividen los datos en fragmentos más pequeños (es decir, el inicio de la corriente de datos es el primer byte del archivo 00000000.evt; el final de la corriente de datos es el último byte del archivo nnnnnnnn.evt).

El tamaño máximo de cada archivo se puede definir también como el número máximo de archivos. Un supervisor de sucesos no dividirá nunca un solo registro de sucesos entre dos archivos. Sin embargo, el supervisor de sucesos puede grabar registros relacionados en dos archivos distintos. Es responsabilidad de la aplicación que utiliza estos datos realizar un seguimiento de dicha información relacionada cuando procese los archivos de sucesos.

nombre-vía

El nombre del directorio en el que el supervisor de sucesos debe grabar los datos de los archivos de sucesos. La vía de acceso debe ser conocida en el servidor, sin embargo, la vía de acceso en sí puede residir en otra partición o nodo (por ejemplo, en un sistema basado en UNIX), puede ser un archivo montado NFS). Se debe utilizar una constante de serie cuando se especifica el *nombre-vía*.

El directorio no tiene que existir en el momento de CREATE EVENT MONITOR. Sin embargo, se realiza una comprobación de la existencia de la vía de acceso de destino cuando se activa el supervisor de sucesos. En este momento, si la vía de acceso de destino no existe, se genera un error (SQLSTATE 428A3).

Si se especifica una vía de acceso absoluta (una vía de acceso que empieza en el directorio raíz de AIX, o un identificador de disco en OS/2, Windows 95 y Windows NT), la vía de acceso especificada será la que se utilice. Si se especifica una vía de acceso relativa (una vía que no empieza en la raíz), se utilizará la vía de acceso al directorio DB2EVENT en el directorio de la base de datos.

Cuando se especifique una vía de acceso relativa, se utiliza el directorio DB2EVENT para convertirla en una vía de acceso absoluta. En adelante, no se distinguirá entre las vías de acceso absoluta y relativa. La vía de acceso absoluta se almacena en la vista de catálogo SYSCAT.EVENTMONITORS.

Es posible especificar dos o más supervisores de sucesos que tengan la misma vía de acceso de destino. Sin embargo, cuando uno de los supervisores de sucesos se ha activado por primera vez, y siempre que el directorio de destino no esté vacío, será imposible activar cualquiera de los supervisores de sucesos.

Opciones de archivo

Especifica las opciones para el formato del archivo.

MAXFILES NONE

Especifica que no hay ningún límite en el número de archivos de sucesos que vaya a crear el supervisor de sucesos. Éste es el valor por omisión.

MAXFILES *número-de-archivos*

Especifica que hay un límite en el número de archivos del supervisor de sucesos que vayan a existir para un supervisor de sucesos en particular en cualquier momento. Siempre que un supervisor de sucesos tenga que crear otro archivo, comprobará que el número de archivos .evt del directorio sea inferior al *número-de-archivos*. Si ya se ha alcanzado este límite, el supervisor de sucesos se desactivará por sí solo.

Si una aplicación elimina los archivos de sucesos del directorio después de haberlos grabado, el número total de archivos que un supervisor de sucesos puede producir

puede exceder el *número-de-archivos*. Esta opción se ha proporcionado para permitir a un usuario garantizar que los datos de sucesos no consumirán más de una cantidad de espacio de disco especificada.

MAXFILESIZE *páginas*

Especifica que hay un límite en el tamaño de cada archivo del supervisor de sucesos. Siempre que un supervisor de sucesos grabe un nuevo registro de suceso en un archivo, comprueba que el archivo no vaya a crecer de manera que sobrepase las *páginas* (en unidades de páginas de 4K). Si el archivo resultante fuese a ser demasiado grande, entonces el supervisor de sucesos conmutará al siguiente archivo.

El valor por omisión para esta opción es:

- OS/2, Windows 95 y Windows NT - 200 páginas de 4K
- UNIX - 1000 páginas de 4K

El número de páginas debe ser mayor que el tamaño del almacenamiento intermedio de sucesos en páginas, como mínimo. Si no se cumple este requisito, se generará un error (SQLSTATE 428A4).

MAXFILESIZE NONE

Especifica que no hay ningún límite establecido en el tamaño de un archivo. Si se especifica MAXFILESIZE NONE, también debe especificarse MAXFILES 1. Esta opción significa que un archivo contendrá todos los datos de sucesos para un supervisor de sucesos en particular. En este caso el único archivo de sucesos será 00000000.evt.

BUFFERSIZE *páginas*

Especifica el tamaño de los almacenamientos intermedios del supervisor de sucesos (en unidades de páginas de 4K). Todas las E/S del archivo del supervisor de sucesos se almacenan temporalmente para mejorar el rendimiento de los supervisores de sucesos. Cuanto más grandes sean los almacenamientos intermedios, menos E/S realizará el supervisor de sucesos. Los supervisores de sucesos altamente activos deben tener almacenamientos intermedios mayores que los supervisores de sucesos relativamente inactivos. Cuando se inicia el supervisor, se asignan dos almacenamientos intermedios del tamaño especificado. Los supervisores de sucesos utilizan el doble de almacenamiento intermedio para permitir E/S asíncronas.

El tamaño mínimo y por omisión de cada almacenamiento intermedio (si no se especifica esta opción) es 4 páginas (es decir, 2 almacenamientos intermedios, cada uno con un tamaño de 16 K). El tamaño máximo de los almacenamientos intermedios está limitado por el tamaño del almacenamiento dinámico del supervisor (MON_HEAP), pues el espacio para los almacenamientos intermedios se asigna a partir del almacenamiento dinámico. Si utiliza simultáneamente muchos supervisores de sucesos, aumente el valor del parámetro de configuración MON_HEAP de la base de datos.

Los supervisores de sucesos que escriben sus datos en una conexión ("pipe") también tienen dos almacenamientos intermedios internos (no configurables) que tienen un tamaño de 1 página cada uno. El espacio para estos almacenamientos intermedios también se asigna a partir del almacenamiento dinámico del supervisor (MON_HEAP). Para cada supervisor de sucesos activo que tiene un destino de conexión, aumente el tamaño del almacenamiento dinámico de la base de datos en 2 páginas.

BLOCKED

Especifica que cada agente que genera un suceso debe esperar a que se grabe en disco un almacenamiento intermedio de sucesos si el agente determina que ambos almacenamientos intermedios de sucesos están llenos. Debe seleccionarse BLOCKED para garantizar que no se van a perder datos. Es la opción por omisión.

NONBLOCKED

Especifica que cada agente que genera un suceso no debe esperar a que se grabe en disco un almacenamiento intermedio de sucesos si el agente determina que ambos almacenamientos intermedios de sucesos están llenos. Los supervisores de sucesos NONBLOCKED no hacen que las operaciones vayan más despacio como los supervisores de sucesos BLOCKED. Sin embargo, los supervisores de sucesos NONBLOCKED están sujetos a la pérdida de datos en sistemas muy activos.

APPEND

Especifica que si ya existen archivos de datos de sucesos cuando se activa el supervisor de sucesos, éste añadirá los nuevos datos de sucesos a los archivos de corriente de datos existentes. Cuando el supervisor de sucesos se

CREATE EVENT MONITOR

reactiva, reanudará la escritura en los archivos de sucesos como si nunca se hubiese desactivado. APPEND es la opción por omisión.

La opción APPEND no se aplica al momento de CREATE EVENT MONITOR, si hay datos de sucesos existentes en el directorio donde el supervisor de sucesos que se acaba de crear va a grabar sus datos de sucesos.

REPLACE

Especifica que si ya existen archivos de datos de sucesos cuando se activa el supervisor de sucesos, éste borrará todos los archivos de sucesos y empezará a grabar los datos en el archivo 00000000.evt.

MANUALSTART

Especifica que el supervisor de sucesos no se iniciará automáticamente cada vez que se inicie la base de datos. Los supervisores de sucesos con la opción MANUALSTART deben activarse manualmente utilizando la sentencia SET EVENT MONITOR STATE. Es la opción por omisión.

AUTOSTART

Especifica que el supervisor de sucesos se iniciará automáticamente cada vez que se inicie la base de datos.

ON NODE

Palabra clave que indica que se especifican particiones específicas.

número-nodo

Especifica un número de partición donde el supervisor de sucesos ejecuta y graba los sucesos. Con el ámbito de supervisión definida como GLOBAL, todas las particiones informan al número de partición especificado. El componente de E/S se ejecutará físicamente en la partición especificada, escribiendo sus registros en el directorio /tmp/dlocks de esa partición.

GLOBAL

El supervisor de sucesos informa de todas la particiones. Para una base de datos particionada en DB2 Universal Database Versión 7, sólo se pueden definir como GLOBAL los supervisores de sucesos de puntos muertos. El supervisor de sucesos global informará de los puntos muertos para todos los nodos del sistema.

LOCAL

El supervisor de sucesos sólo informa sobre la partición que está ejecutando. Ofrece un rastreo parcial de la actividad de la base de datos. Éste es el valor por omisión.

Normas

- Cada uno de los tipos de sucesos (DATABASE, TABLES, DEADLOCK,...) sólo pueden especificarse una vez en la definición de un supervisor de sucesos en particular.

Notas

- Las definiciones del supervisor de sucesos se registran en la vista de catálogo SYSCAT.EVENTMONITORS. Los sucesos en sí se registran en la vista de catálogo SYSCAT.EVENTS.
- Para obtener información detallada sobre la utilización del supervisor de la base de datos y sobre la interpretación de datos de las conexiones y archivos, consulte el manual *System Monitor Guide and Reference*.

Ejemplos

Ejemplo 1: El siguiente ejemplo crea un supervisor de sucesos denominado SMITHPAY. Este supervisor de sucesos, reunirá los datos de sucesos para la base de datos así como para las sentencias de SQL realizadas por la aplicación PAYROLL propiedad del ID de autorización JSMITH. Los datos se añadirán a la vía de acceso absoluta /home/jsmith/event/smithpay/. Se crearán un máximo de 25 archivos. Cada archivo tendrá una longitud máxima de 1 024 páginas de 4K. La E/S del archivo no estará bloqueada.

```
CREATE EVENT MONITOR SMITHPAY
  FOR DATABASE, STATEMENTS
  WHERE APPL_NAME = 'PAYROLL' AND AUTH_ID = 'JSMITH'
  WRITE TO FILE '/home/jsmith/event/smithpay'
  MAXFILES 25
  MAXFILESIZE 1024
  NONBLOCKED
  APPEND
```

Ejemplo 2: El siguiente ejemplo crea un supervisor de sucesos denominado DEADLOCKS_EVTs. Este supervisor de sucesos reunirá los sucesos de puntos muertos y los grabará en la vía de acceso relativa DLOCKS. Se grabará un archivo y no hay tamaño de archivo máximo. Cada vez que se active el supervisor de sucesos, se añadirán los datos de sucesos al archivo 00000000.evt si existe. El supervisor de sucesos se iniciará cada vez que se inicie la base de datos. La E/S estará bloqueada por omisión.

```
CREATE EVENT MONITOR DEADLOCK_EVTs
  FOR DEADLOCKS
  WRITE TO FILE 'DLOCKS'
  MAXFILES 1
  MAXFILESIZE NONE
  AUTOSTART
```

Ejemplo 3: Este ejemplo crea un supervisor de sucesos denominado DB_APPLs. Este supervisor de sucesos reúne sucesos de conexión y graba datos en la conexión con nombre /home/jsmith/applpipe.

CREATE EVENT MONITOR

```
CREATE EVENT MONITOR DB_APPLS  
FOR CONNECTIONS  
WRITE TO PIPE '/home/jsmith/applpipe'
```

CREATE FUNCTION

Esta sentencia se utiliza para registrar o definir una función definida por el usuario o una plantilla de función en un servidor de aplicaciones.

Existen cinco tipos diferentes de funciones que se pueden crear utilizando esta sentencia. Cada una de ellas se describe por separado.

- Escalar externa

La función se escribe en un lenguaje de programación y devuelve un valor escalar. El ejecutable externo se registra en la base de datos junto con diversos atributos de la función. Vea “CREATE FUNCTION (Escalar externa)” en la página 696.

- Tabla externa

La función se escribe en un lenguaje de programación y devuelve una tabla completa. El ejecutable externo se registra en la base de datos junto con diversos atributos de la función. Vea “CREATE FUNCTION (Tabla externa)” en la página 723.

- Tabla externa OLE DB

Una función de tabla externa OLE DB definida por el usuario está registrada en la base de datos para acceder a los datos de un proveedor OLE DB. Vea “CREATE FUNCTION (Tabla externa OLE DB)” en la página 741.

- Fuente o plantilla

Una función fuente se implanta invocando otra función (incorporada, externa, SQL o fuente) que ya está registrada en la base de datos. Vea “CREATE FUNCTION (fuente o plantilla)” en la página 750.

Es posible crear una función parcial, denominada *plantilla de función*, que defina qué tipos de valores deben devolverse pero que no contengan ningún código ejecutable. El usuario la correlaciona con una función fuente de datos dentro de un sistema federado, de esta forma se puede invocar la función fuente de datos desde una base de datos federada. Una plantilla de función sólo se puede registrar en un servidor de aplicaciones que esté designado como servidor federado.

- SQL, escalar, de tabla o de fila

El cuerpo de la función está escrito en SQL y se define junto con el registro en la base de datos. Devuelve un valor escalar, una tabla o una fila individual. Vea “CREATE FUNCTION (SQL, escalar, de tabla o de fila)” en la página 761.

CREATE FUNCTION (Escalar externa)

CREATE FUNCTION (Escalar externa)

Esta sentencia se utiliza para registrar una función escalar externa definida por el usuario en un servidor de aplicaciones. Una *función escalar* devuelve un solo valor cada vez que se invoca y en general es válida donde una expresión SQL sea válida

Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse dinámicamente. Sin embargo, si se aplica la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

Autorización

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema de la función no hace referencia a un esquema existente.
- Privilegio CREATEIN para el esquema, si el nombre de esquema de la función no hace referencia a un esquema existente.

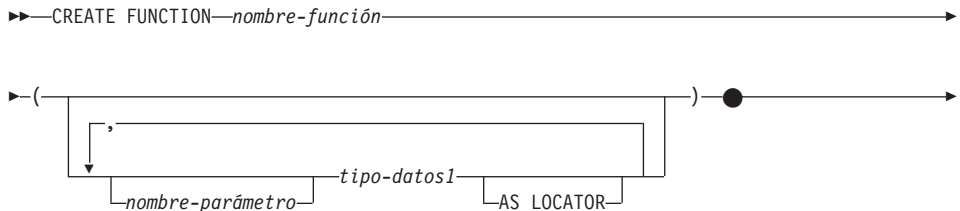
Para crear una función no restringida, el ID de autorización de la sentencia debe tener como mínimo uno de los privilegios siguientes:

- Autorización CREATE_NOT_FENCED para la base de datos
- Autorización SYSADM o DBADM.

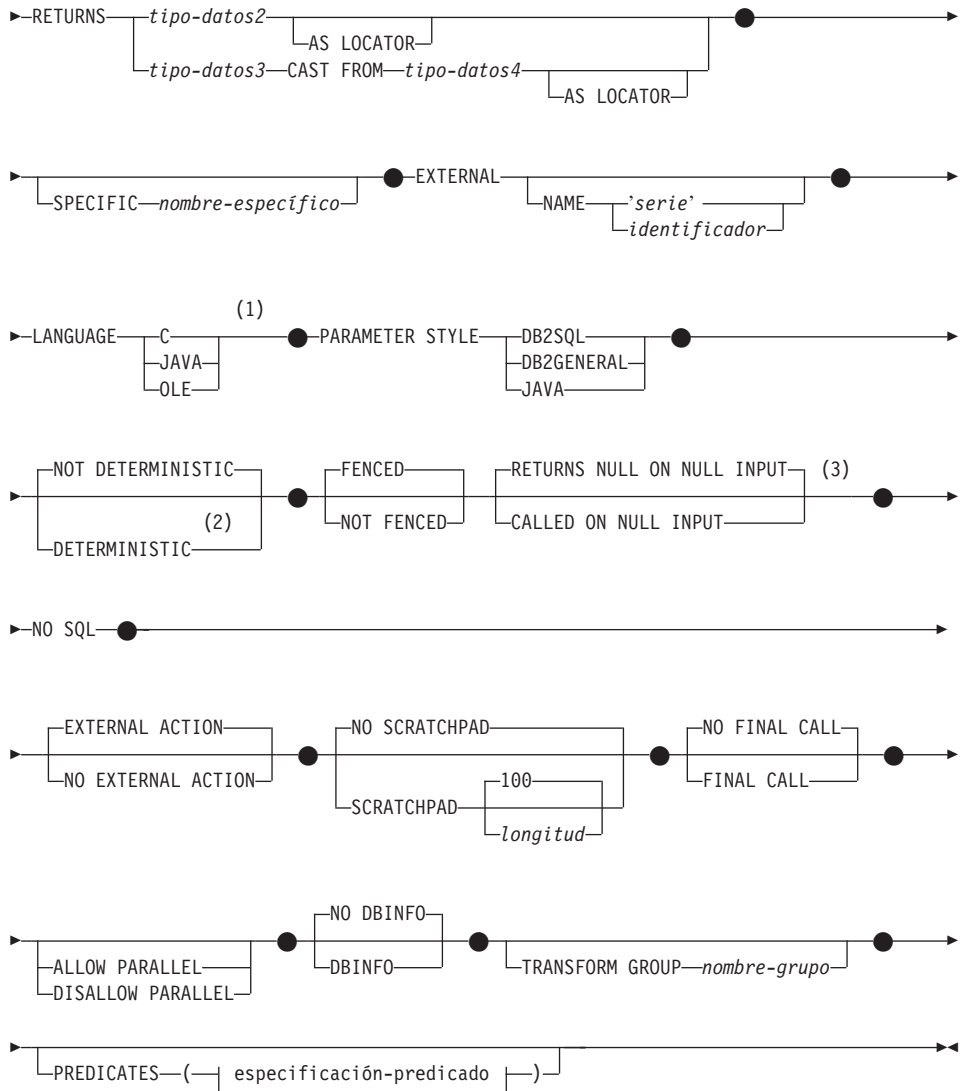
Para crear una función restringida, no se necesitan autorizaciones ni privilegios adicionales.

Si el ID de autorización no tiene la autorización suficiente para realizar la operación, se produce un error (SQLSTATE 42502).

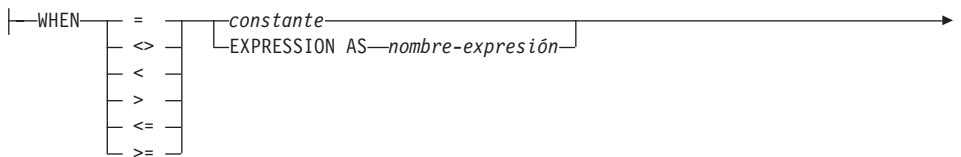
Sintaxis



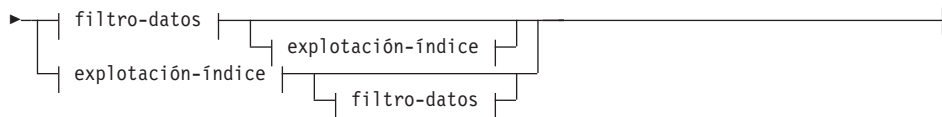
CREATE FUNCTION (Escalar externa)



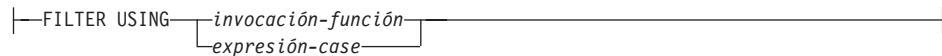
especificación-predicado:



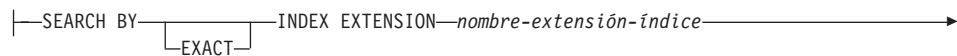
CREATE FUNCTION (Escalar externa)



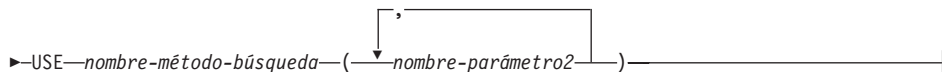
filtro-datos:



explotación-índice:



regla-explotación:



Notas:

- 1 También se da soporte a LANGUAGE SQL. Vea "CREATE FUNCTION (SQL, escalar, de tabla o de fila)" en la página 761.
- 2 Puede especificarse NOT VARIANT en lugar de DETERMINISTIC y VARIANT puede especificarse en lugar de NOT DETERMINISTIC.
- 3 Puede especificarse NULL CALL en lugar de CALLED ON NULL INPUT y puede especificarse NOT NULL CALL en lugar de RETURNS NULL ON NULL INPUT.

Descripción

nombre-función

Indica el nombre de la función que se está definiendo. Consiste en el nombre, calificado o no calificado, que designa una función. La forma no calificada del *nombre-función* es un identificador SQL (cuya longitud

máxima es de 18). En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace QUALIFIER especifica de forma implícita el calificador para los nombres de objeto no calificados. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL. El nombre calificado no debe ser el mismo que el tipo de datos del primer parámetro, si ese parámetro es un tipo estructurado.

El nombre, incluidos los calificadores implícitos y explícitos, junto con el número de parámetros y el tipo de datos de cada parámetro (sin tener en cuenta ningún atributo de longitud, precisión o escala del tipo de datos) no debe identificar una función o método descritos en el catálogo (SQLSTATE 42723). El nombre no calificado, junto con el número y el tipo de datos de los parámetros, que por supuesto es exclusivo en su esquema, no es necesario que lo sea en todos los esquemas.

Si se especifica un nombre compuesto de dos partes, el *nombre-esquema* no puede empezar por "SYS". De lo contrario, se genera un error (SQLSTATE 42939).

Hay algunos nombres que se utilizan como palabras clave en predicados que están reservados para que los utilice el sistema y no pueden utilizarse como *nombre-función*. Estos nombres son SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH y los operadores de comparación, tal como se describen en el apartado "Predicado básico" en la página 213. Cualquier incumplimiento de esta regla provocará la aparición de un error (SQLSTATE 42939).

En general, puede utilizarse el mismo nombre para más de una función si existe alguna diferencia en la signatura de las funciones.

Aunque no hay ninguna prohibición al respecto, una función de tabla externa definida por el usuario no debe tener el mismo nombre que una función incorporada, a menos que sea una alteración temporal intencionada. Si se otorga a una función que tiene un significado diferente el mismo nombre (por ejemplo, LENGTH, VALUE, MAX), con argumentos coherentes (caso de una función escalar incorporada o una función de columna), se corre el riesgo de provocar errores en las sentencias del SQL dinámico o al volver a enlazar las aplicaciones del SQL estático; es posible que la aplicación falle, o incluso peor, que parezca que se ejecuta satisfactoriamente y genere un resultado diferente.

nombre-parámetro

Designa el parámetro que se puede utilizar en la definición de función subsiguiente. Los nombres de parámetro son necesarios para hacer referencia a los parámetros de una función en la cláusula de *plotación-índice* de una especificación de predicado.

CREATE FUNCTION (Escalar externa)

(tipo-datos1,...)

Identifica el número de parámetros de entrada de la función, al tiempo que especifica el tipo de datos de cada parámetro. En la lista debe especificarse una entrada por cada parámetro que la función espera recibir. No se permiten más de 90 parámetros. En caso de sobrepasarse este límite, se produce un error (SQLSTATE 54023).

Existe la posibilidad de registrar una función que no tenga ningún parámetro. En este caso, sigue siendo necesario escribir los paréntesis, sin ningún tipo de datos interpuesto. Por ejemplo,

```
CREATE FUNCTION WOOFER() ...
```

En un esquema, no se permite que dos funciones que se denominen igual tengan exactamente el mismo tipo para todos los parámetros correspondientes. Las longitudes, precisiones y escalas no se tienen en cuenta en esta comparación de tipos. Por esta razón, se considera que CHAR(8) y CHAR(35) tienen el mismo tipo, que son DECIMAL(11,2) y DECIMAL(4,3). Hay una agrupación más profunda de los tipos que hace que se traten como el mismo tipo para esta finalidad como, por ejemplo, DECIMAL y NUMERIC. Una signatura duplicada produce un error de SQL (SQLSTATE 42723).

Por ejemplo, si se emiten estas sentencias:

```
CREATE FUNCTION PART (INT, CHAR(15)) ...  
CREATE FUNCTION PART (INTEGER, CHAR(40)) ...  
  
CREATE FUNCTION ANGLE (DECIMAL(12,2)) ...  
CREATE FUNCTION ANGLE (DEC(10,7)) ...
```

la segunda sentencia y la cuarta fallarían porque se considera que son funciones duplicadas.

tipo-datos1

Especifica el tipo de datos del parámetro.

- Pueden proporcionarse especificaciones y abreviaturas de tipo de datos SQL que puedan especificarse en la definición de *tipo-datos1* de la sentencia CREATE TABLE y que tengan una correspondencia en el lenguaje que se utiliza para escribir la función. Consulte las secciones específicas de los lenguajes del manual *Application Development Guide* para obtener detalles sobre la correlación entre los tipos de datos SQL y los tipos de datos del lenguaje principal con respecto a las funciones definidas por el usuario.
- DECIMAL (y NUMERIC) no son válidos con LANGUAGE C y OLE (SQLSTATE 42815). Para buscar alternativas a la utilización de DECIMAL consulte el manual *Application Development Guide*.

CREATE FUNCTION (Escalar externa)

- REF(*nombre-tipo*) puede especificarse como tipo de datos de un parámetro. Sin embargo, dicho parámetro no debe tener ámbito.
- Se pueden especificar tipos estructurados, a condición de que existan las funciones de transformación apropiadas en el grupo de transformación asociado.

AS LOCATOR

Para los tipos LOB o los tipos diferenciados que se basan en un tipo LOB, se puede añadir la cláusula AS LOCATOR. Esto indica que se ha de pasar un localizador de LOB a la UDF en lugar del valor real. Esto reduce considerablemente el número de bytes que se pasan a la UDF y puede también mejorar el rendimiento, especialmente cuando la UDF sólo necesite unos pocos bytes. La utilización de los localizadores de LOB en las UDF se describen en el manual *Application Development Guide*.

El siguiente ejemplo ilustra la utilización de la cláusula AS LOCATOR en las definiciones de parámetros:

```
CREATE FUNCTION foo (CLOB(10M) AS LOCATOR, IMAGE AS LOCATOR)
...
```

lo que supone que IMAGE es un tipo diferenciado basado en uno de los tipos LOB.

Observe también que para fines de promoción de argumentos, la cláusula AS LOCATOR no tiene ningún efecto. En el ejemplo, los tipos se consideran CLOB e IMAGE respectivamente, lo que significa que podría pasarse un argumento CHAR o VARCHAR a la función como el primer argumento. Igualmente, AS LOCATOR no tiene ningún efecto en la signatura de la función, que se utiliza en la comparación de la función (a) cuando se hace referencia en DML, por un proceso llamado "resolución de función" y (b) cuando se hace referencia en una sentencia DDL como, por ejemplo, COMMENT ON o DROP. De hecho, la cláusula puede utilizarse o no en una sentencia COMMENT ON o DROP sin que tenga ningún significado.

Se produce un error (SQLSTATE 42601) si se especifica AS LOCATOR para un tipo que no sea LOB o para un tipo diferenciado basado en un LOB.

Si la función es FENCED, no se puede especificar la cláusula AS LOCATOR (SQLSTATE 42613).

RETURNS

Esta cláusula obligatoria identifica el resultado de la función.

CREATE FUNCTION (Escalar externa)

tipo-datos2

Especifica el tipo de datos de la salida.

En este caso, se aplican exactamente las mismas consideraciones que para los parámetros de funciones externas descritas arriba bajo *tipo-datos1* para parámetros de función.

AS LOCATOR

Para tipos LOB o tipos diferenciados que están basados en tipos LOB, se puede añadir la cláusula AS LOCATOR. Esto indica que se ha de pasar un localizador de LOB de la UDF en lugar del valor real.

tipo-datos3 **CAST FROM** *tipo-datos4*

Especifica el tipo de datos de la salida.

Este formato de cláusula RETURNS se utiliza para devolver un tipo de datos diferente a la sentencia de invocación del tipo de datos que se ha devuelto por el código de función. Por ejemplo, en

```
CREATE FUNCTION GET_HIRE_DATE(CHAR(6))  
  RETURNS DATE CAST FROM CHAR(10)  
  ...
```

el código de función devuelve un valor CHAR(10) al gestor de bases de datos que, a su vez, lo convierte en DATE y pasa dicho valor a la sentencia que realiza la invocación. El *tipo-datos4* debe ser convertible al parámetro *tipo-datos3*. Si no es convertible, se produce un error (SQLSTATE 42880) (vea "Conversión entre tipos de datos" en la página 101 para conocer la definición de "convertible").

Debido a que la longitud, precisión o escala de *tipo-datos3* puede inferirse de *tipo-datos4*, no es necesario (pero está permitido) especificar la longitud, precisión o escala de los tipos parametrizados especificados para *tipo-datos3*. En su lugar, pueden utilizarse paréntesis vacíos (por ejemplo, puede utilizarse VARCHAR()). No puede utilizarse FLOAT() (SQLSTATE 42601) puesto que el valor del parámetro indica tipos de datos diferentes (REAL o DOUBLE).

No pueden especificarse tipos diferenciados ni tipos estructurados como tipo para *tipo-datos4* (SQLSTATE 42815).

La operación de conversión del tipo de datos también está sujeta a comprobaciones durante la ejecución que pueden dar lugar a errores de conversión.

AS LOCATOR

Para especificaciones de *tipo-datos4* que sean tipos LOB o tipos diferenciados que estén basados en tipos LOB, se puede añadir la cláusula AS LOCATOR. Esto indica que se ha de devolver un

CREATE FUNCTION (Escalar externa)

localizador de LOB de una UDF en lugar del valor actual. La utilización de los localizadores de LOB en las UDF se describen en el manual *Application Development Guide*.

SPECIFIC *nombre-específico*

Proporciona un nombre exclusivo para la instancia de la función que se está definiendo. Este nombre específico puede utilizarse cuando se utiliza esta función como fuente, al eliminar la función o bien al comentarla. Nunca puede emplearse para invocar a la función. La forma no calificada de *nombre-específico* es un identificador SQL (cuya longitud máxima es 18). La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL. El nombre (incluido el calificador implícito o explícito) no debe identificar otra instancia de función ni especificación de método que exista en el servidor de aplicaciones; de lo contrario, se produce un error (SQLSTATE 42710).

El *nombre-específico* puede ser el mismo que un *nombre-función* ya existente.

Si no se especifica ningún calificador, se emplea el que se haya utilizado para el *nombre-función*. Si se especifica un calificador, éste debe ser el mismo que el calificador explícito o implícito del *nombre-función*; de lo contrario se produce un error (SQLSTATE 42882).

Si no se especifica el *nombre-específico*, el gestor de bases de datos genera un nombre exclusivo. El nombre exclusivo es SQL seguido de una indicación de fecha y hora en forma de serie caracteres: SQLaammdhhmssxxx.

EXTERNAL

Esta cláusula indica que la sentencia CREATE FUNCTION se emplea para registrar una nueva función basada en el código escrito en un lenguaje de programación externo y cumple los convenios estipulados para los enlaces e interfaces.

Si no se especifica la cláusula NAME se supone "NAME *nombre-función*".

NAME '*serie*'

Esta cláusula identifica el nombre del código escrito por el usuario que implanta la función que se está definiendo.

La opción '*serie*' es una constante de serie con un máximo de 254 caracteres. El formato que se utiliza para la serie depende de la opción LANGUAGE especificada.

- Para LANGUAGE C:

La *serie* especificada constituye el nombre de la biblioteca y la función de la biblioteca que el gestor de bases de datos invoca para ejecutar la función definida por el usuario que se está creando (CREATE). Al ejecutar la sentencia CREATE FUNCTION, no es preciso que exista la biblioteca (ni la función dentro de la

CREATE FUNCTION (Escalar externa)

biblioteca). Sin embargo, cuando se utiliza la función en una sentencia de SQL, la biblioteca y la función de la biblioteca deben existir y ser accesibles desde la máquina servidora de bases de datos, de lo contrario se produce un error (SQLSTATE 42724).

```
► '—id_biblioteca—', —id_vía_absoluta—, —id_func—' ►
```

Dentro de los apóstrofes no puede haber ningún espacio en blanco adicional.

id_biblioteca

Identifica el nombre de la biblioteca que contiene la función. El gestor de bases de datos buscará la biblioteca en el directorio `.../sqllib/function` (sistemas basados en UNIX), o en el directorio `...\nombre_instancia\function` (OS/2, y Sistemas operativos Windows de 32 bits según lo especificado por la variable de registro DB2INSTPROF), donde el gestor de bases de datos localizará el directorio sqllib de control que se utiliza para ejecutar el gestor de bases de datos. Por ejemplo, el directorio de control sqllib en los sistemas basados en UNIX es `/u/$DB2INSTANCE/sqllib`.

Si `'mifunc'` es *library_id* en un sistema basado en UNIX, hace que el gestor de bases de datos busque la función en la biblioteca `/u/production/sqllib/function/mifunc`, siempre que el gestor de bases de datos se ejecute desde `/u/production`.

En OS/2 y Sistemas operativos Windows de 32 bits, el gestor de bases de datos buscará en LIBPATH o PATH si el *id_biblioteca* no está ubicado en el directorio de la función.

En OS/2, el *id_biblioteca* no debe contener más de 8 caracteres.

id_vía_absoluta

Identifica el nombre completo de vía de acceso del archivo que contiene la función.

En un sistema basado en UNIX, por ejemplo, `'/u/jchui/mibib/mifunc'` hace que el gestor de bases de datos busque en `/u/jchui/mibib` la biblioteca compartida `mifunc`.

En el OS/2 y Sistemas operativos Windows de 32 bits, `'d:\mibib\mifunc'` hace que el gestor de bases de datos cargue la biblioteca de enlace dinámico, el archivo `mifunc.dll`, desde el directorio `d:\mibib`.

En OS/2, la última parte de esta especificación (es decir, el nombre de la dll), no debe contener más de 8 caracteres.

CREATE FUNCTION (Escalar externa)

! id_func

Identifica el nombre del punto de entrada de la función que ha de invocarse. El signo ! sirve como delimitador entre el ID de biblioteca y el ID de función. Si se omite *! id_func*, el gestor de bases de datos utilizará el valor que había por omisión para el punto de entrada cuando se ha enlazado la biblioteca.

En un sistema basado en UNIX, por ejemplo, 'mimod!func8' dirige el gestor de bases de datos para que busque la biblioteca \$inst_home_dir/sqllob/function/mimod y utilice el punto de entrada func8 dentro de dicha biblioteca.

En el OS/2 y Sistemas operativos Windows de 32 bits, 'mimod!func8' hace que el gestor de bases de datos cargue el archivo mimod.dll y llame a la función func8() de la biblioteca de enlace dinámico (DLL).

Si la composición de la serie de caracteres no es correcta, se produce un error (SQLSTATE 42878).

El cuerpo de cada función externa debe estar en un directorio que sea accesible en todas las particiones de la base de datos.

- Para LANGUAGE JAVA:

La *serie* especificada contiene el identificador opcional del archivo jar, el identificador de clase y el identificador de método, que el gestor de bases de datos invoca para ejecutar la función definida por el usuario que se está creando. No es necesario que existan el identificador de clase ni el identificador de método cuando se ejecuta la sentencia CREATE FUNCTION. Si se especifica un *id_jar*, éste debe existir cuando se ejecuta la sentencia CREATE FUNCTION. Sin embargo, cuando se utiliza la función en una sentencia de SQL, debe existir el identificador de método y debe ser accesible desde la máquina servidora de bases de datos, de lo contrario se produce un error (SQLSTATE 42724).

► ' *id_clase* *id_método* ' ◀

 └─ jar_id : ─┘ └─┘

Dentro de los apóstrofes no puede haber ningún espacio en blanco adicional.

id_jar

Designa el identificador de jar que se asignó a la colección jar cuando se instaló en la base de datos. Puede ser un identificador simple o un identificador calificado por un esquema. Algunos ejemplos son 'miJar' y 'miEsquema.miJar'.

CREATE FUNCTION (Escalar externa)

id_clase

Identifica el identificador de clase del objeto Java. Si la clase forma parte de un paquete, la parte del identificador de clase debe incluir el prefijo completo del paquete, por ejemplo 'miPacks.UserFuncs'. La máquina virtual Java buscará en el directorio '.../miPacks/UserFuncs/' correspondiente a las clases. En OS/2 y Sistemas operativos Windows de 32 bits, la máquina virtual Java buscará en el directorio '...\miPacks\UserFuncs\.'

id_método

Identifica el nombre de método del objeto Java que se invoca.

- Para LANGUAGE OLE:

La *serie* especificada es el identificador del programa OLE (*idprog*) o identificador de clase (*idcls*) y el identificador del método, que invoca el gestor de bases de datos para ejecutar la función definida por el usuario que se está creando (CREATE). No es preciso que exista el identificador de programa ni el identificador de clase y el identificador de método al emitir la sentencia CREATE FUNCTION. No obstante, cuando se utiliza la función en una sentencia de SQL, debe existir el identificador de método y debe ser accesible desde la máquina servidora de bases de datos, de lo contrario se produce un error (SQLSTATE 42724).

► '—*idprog*—|—*id_método*—' ◄
└──┬──┘
 idcls

Dentro de los apóstrofes no puede haber ningún espacio en blanco adicional.

idprog

Identifica el identificador de programa del objeto OLE.

idprog no se interpreta por el gestor de bases de datos sino que sólo se reenvía a la API OLE en tiempo de ejecución. El objeto OLE especificado debe poderse crear y dar soporte a un enlace lógico posterior (denominado también enlace lógico basado en IDispatch).

idcls

Designa el identificador de clase del objeto OLE que se debe crear. Puede utilizarse como una alternativa para especificar *idprog* en el caso de que el objeto OLE no esté registrado con un *idprog*. El *idcls* tiene el formato:

{nnnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnnn}

CREATE FUNCTION (Escalar externa)

donde 'n' es un carácter alfanumérico. *idcls* no se interpreta por el gestor de bases de datos, sólo se reenvía a las API OLE en el momento de la ejecución.

id_método

Identifica el nombre de método del objeto OLE que se ha de invocar.

NAME *identificador*

Este *identificador* especificado es un identificador SQL. El identificador SQL se utiliza como el *id-biblioteca* en la serie. A menos que sea un identificador delimitado, se convierte a mayúsculas. Si el identificador está calificado con un nombre de esquema, se ignora la parte correspondiente al nombre de esquema. Este formato de NAME sólo puede utilizarse con LANGUAGE C.

LANGUAGE

Esta cláusula obligatoria se emplea para especificar el convenio de la interfaz de lenguaje en el que se está escrito el cuerpo de la función definida por el usuario.

C Esto significa que el gestor de bases de datos llamará a la función definida por el usuario como si se tratase de una función en C. La función definida por el usuario debe ajustarse al convenio de llamadas y enlaces del lenguaje C, tal y como se define en el prototipo de C estándar de ANSI.

JAVA Esto significa que el gestor de bases de datos llamará a la función definida por el usuario como un método en una clase Java.

OLE Significa que el gestor de bases de datos llamará a la función definida por el usuario como si fuese un método expuesto por un objeto de automatización OLE. La función definida por el usuario debe ajustarse a los tipos de datos de automatización OLE y al mecanismo de invocación, tal como se describe en el manual *OLE Automation Programmer's Reference*.

Sólo se da soporte al LANGUAGE OLE para las funciones definidas por el usuario almacenadas en DB2 para Sistemas operativos Windows de 32 bits.

PARAMETER STYLE

Esta cláusula se utiliza para especificar los convenios utilizados para pasar parámetros a funciones y devolver el valor de las mismas.

DB2SQL

Se utiliza para especificar los convenios para pasar parámetros a, y devolver el valor de, las funciones externas que cumplen con los convenios de enlace y de llamada del lenguaje C o los métodos

CREATE FUNCTION (Escalar externa)

expuestos por los objetos de automatización OLE. Debe especificarse cuando se utiliza LANGUAGE C o LANGUAGE OLE.

DB2GENERAL

Se utiliza para especificar los convenios para pasar parámetros a, y devolver el valor de, las funciones externas que están definidas como un método en una clase Java. Esto sólo puede especificarse cuando se utiliza LANGUAGE JAVA.

El valor DB2GENRL puede utilizarse como sinónimo de DB2GENERAL.

JAVA Significa que la función utilizará un convenio para pasar parámetros que se ajusta a la especificación de lenguaje Java y Rutinas SQLJ. Esto sólo puede especificarse cuando se utiliza LANGUAGE JAVA y no existen tipos estructurados como tipos de parámetro o de retorno (SQLSTATE 429B8). Las funciones PARAMETER STYLE JAVA no dan soporte a las cláusulas FINAL CALL, SCRATCHPAD ni DBINFO.

Consulte la publicación *Application Development Guide* para obtener detalles sobre cómo pasar parámetros.

DETERMINISTIC o NOT DETERMINISTIC

Esta cláusula opcional especifica si la función siempre devuelve los mismos resultados para unos valores argumento determinados (DETERMINISTIC) o si la función depende de ciertos valores de estado que afectan a los resultados (NOT DETERMINISTIC). Es decir, una función DETERMINISTIC siempre debe devolver el mismo resultado para invocaciones sucesivas con entradas de datos idénticas. Mediante la especificación de NOT DETERMINISTIC, se evitan las optimizaciones que aprovechan el hecho de que las entradas idénticas siempre produzcan los mismos resultados. Un ejemplo de una función NOT DETERMINISTIC sería un generador de números aleatorios. Un ejemplo de una función DETERMINISTIC sería una función que determinara la raíz cuadrada de los datos de entrada.

FENCED o NOT FENCED

Esta cláusula especifica si la función se considera o no “segura” para ejecutarse en el proceso o espacio de direcciones del entorno operativo del gestor de bases de datos (NOT FENCED) o no (FENCED).

Si una función se registra como FENCED, el gestor de bases de datos aísla los recursos internos de base de datos (por ejemplo, almacenamientos intermedios de datos) para que la función no pueda acceder a ellos. La mayoría de las funciones tienen la opción de ejecutarse como FENCED o

CREATE FUNCTION (Escalar externa)

NOT FENCED. En general, una función que se ejecute como FENCED no funcionará tan bien como otra de iguales características que se ejecute como NOT FENCED.

Aviso: La utilización de NOT FENCED para funciones no codificadas correctamente puede comprometer la integridad de DB2. DB2 toma algunas precauciones para muchas de las anomalías más habituales que podrían producirse, pero no puede asegurar la integridad completa si se emplean funciones definidas por el usuario como NOT FENCED.

Observe que, aunque el uso de FENCED protege la integridad de la base de datos mejor que NOT FENCED, una UDF definida como FENCED que no se haya codificado, revisado y probado debidamente puede también causar una anomalía involuntaria de DB2.

Normalmente, la mayoría de las funciones definidas por el usuario se pueden ejecutar como FENCED o NOT FENCED. Una función definida con LANGUAGE OLE sólo admite la especificación FENCED (SQLSTATE 42613).

Si la función es FENCED, no se puede especificar la cláusula AS LOCATOR (SQLSTATE 42613).

Para cambiar de FENCED a NOT FENCED, se debe volver a registrar la función (eliminandola y, a continuación, volviendo a crearla). Para que una función definida por el usuario se pueda registrar como NOT FENCED es necesaria la autorización SYSADM, la autorización DBADM o una autorización especial (CREATE_NOT_FENCED).

RETURNS NULL ON NULL INPUT o CALLED ON NULL INPUT

Esta cláusula opcional puede utilizarse para evitar una llamada a la función externa si cualquiera de los argumentos es nulo. Si la función definida por el usuario está definida para no tener ningún parámetro, entonces por supuesto, esta condición de argumento nulo no puede surgir y no importa cómo se haya codificado esta especificación.

Si se especifica RETURNS NULL ON NULL INPUT y, durante la ejecución, cualquiera de los argumentos de la función es nulo, no se invoca la función definida por el usuario y el resultado es el valor nulo.

Si se especifica CALLED ON NULL INPUT, entonces con independencia de si los argumentos son nulos o no, se invoca la función definida por el usuario. Puede devolver un valor nulo o un valor normal (no nulo). Pero corresponde a la UDF comprobar si los valores de los argumentos son nulos.

CREATE FUNCTION (Escalar externa)

El valor NULL CALL puede utilizarse como sinónimo de CALLED ON NULL INPUT para mantener la compatibilidad con versiones anteriores. De manera similar, puede utilizarse NOT NULL CALL como sinónimo de RETURNS NULL ON NULL INPUT.

NO SQL

Esta cláusula obligatoria indica que la función no puede emitir ninguna sentencia de SQL. Si las emite, se produce un error (SQLSTATE 38502) en tiempo de ejecución.

NO EXTERNAL ACTION o EXTERNAL ACTION

Esta cláusula opcional especifica si la función realiza alguna acción que cambia el estado de un objeto no gestionado por el gestor de bases de datos. Para evitar las optimizaciones basadas en que la función no produce ningún efecto externo, se especifica EXTERNAL ACTION. Por ejemplo: enviar un mensaje, hacer sonar una alarma o grabar un registro en un archivo.

NO SCRATCHPAD o SCRATCHPAD *longitud*

Esta cláusula opcional especifica si debe proporcionarse una memoria de trabajo para una función externa. (Es muy recomendable que las funciones definidas por el usuario sean reentrantes, por lo que una memoria de trabajo proporciona un medio para que la función “guarde el estado” entre una llamada y la siguiente.)

Si se especifica SCRATCHPAD, cuando se efectúa la primera invocación de la función definida por el usuario, se asigna memoria para que la función externa utilice una memoria de trabajo. Esta memoria de trabajo tiene las siguientes características:

- *longitud*, si se especifica, define el tamaño en bytes de la memoria de trabajo; este valor debe estar comprendido entre 1 y 32.767 (SQLSTATE 42820). El tamaño por omisión es 100 bytes.
- El valor de inicialización consta sólo de ceros hexadecimales.
- Su ámbito es la sentencia de SQL. Existe una memoria de trabajo para referencia a la función externa en la sentencia de SQL. Por tanto, si la función UDFX de la sentencia siguiente se define con la palabra clave SCRATCHPAD, se asignarían tres memorias de trabajo.

```
SELECT A, UDFX(A) FROM TABLEB
WHERE UDFX(A) > 103 OR UDFX(A) < 19
```

Si se especifica ALLOW PARALLEL o se toma por omisión, el ámbito es diferente del anterior. Si la función se ejecuta en múltiples particiones, se asigna una memoria de trabajo en cada partición donde se procesa la función, para cada referencia a la función en la sentencia de SQL. De manera similar, si la consulta se ejecuta con el paralelismo de intrapartición habilitado, pueden asignarse más de tres memorias de trabajo.

CREATE FUNCTION (Escalar externa)

- Su contenido se conserva. Se conserva el contenido de una llamada de función externa a la llamada siguiente. Los cambios hechos en la memoria de trabajo por la función externa en una llamada seguirán allí en la llamada siguiente. El gestor de bases de datos inicializa las memorias de trabajo al principio de la ejecución de cada sentencia de SQL. El gestor de bases de datos puede restaurar las memorias de trabajo al comienzo de la ejecución de cada subconsulta. El sistema emite una llamada final antes de restaurar una memoria de trabajo si se especifica la opción FINAL CALL.
- Puede utilizarse como punto central de los recursos del sistema (por ejemplo, la memoria) que podría adquirir la función externa. La función podría adquirir la memoria en la primera llamada, conservar su dirección en la memoria de trabajo y acceder a ella en llamadas posteriores.

(En el caso en que se adquiere el recurso del sistema, también debe especificarse la palabra clave FINAL CALL; esto provoca una llamada especial al fin de sentencia dirigida a permitir que la función externa libere cualquier recurso del sistema adquirido.)

Si se especifica SCRATCHPAD, en cada invocación de la función definida por el usuario se pasa un argumento adicional a la función externa que indica la dirección de la memoria de trabajo.

Si se especifica NO SCRATCHPAD, no se asignará ni pasará ninguna memoria de trabajo a la función externa.

SCRATCHPAD no puede utilizarse para funciones PARAMETER STYLE JAVA.

NO FINAL CALL o FINAL CALL

Esta cláusula opcional especifica si debe realizarse una llamada final a una función externa. La finalidad de la misma es hacer que la función externa libere los recursos del sistema que haya adquirido. Junto con la palabra clave SCRATCHPAD, puede ser útil en situaciones donde la función externa adquiera recursos del sistema tales como memoria y los fije en la memoria de trabajo. Si se especifica FINAL CALL, durante la ejecución se produce lo siguiente:

- Se pasa un argumento adicional a la función externa que especifica el tipo de llamada. Los tipos de llamada son:
 - Llamada normal: Se pasan los argumentos SQL y se espera la devolución de un resultado.
 - Primera llamada: la primera llamada a la función externa para esta referencia a la función definida por el usuario en esta sentencia de SQL. La primera llamada es una llamada normal.

CREATE FUNCTION (Escalar externa)

- Llamada final: una llamada final a la función externa para permitir que la función libere recursos. La llamada final no es una llamada normal. Esta llamada final tiene lugar en las siguientes circunstancias:
 - Fin de sentencia: Este caso se produce cuando se cierra el cursor para las sentencias orientadas a cursor o cuando la sentencia termina la ejecución de otra manera.
 - Fin de transacción: Este caso se produce cuando no se da un fin de sentencia normal. Por ejemplo, cuando por alguna razón la lógica de una aplicación ignora el cierre del cursor.

Si se produce una operación de confirmación mientras está abierto un cursor definido como WITH HOLD, se realiza una llamada final en el cierre subsiguiente del cursor o al final de la aplicación.

Si no se especifica NO FINAL CALL, no se pasa ningún argumento de “tipo de llamada” a la función externa y no se realiza ninguna llamada final.

En *Application Development Guide* se incluye una descripción del proceso UDF escalar de estas llamadas cuando se producen errores.

FINAL CALL no recibe soporte para funciones PARAMETER STYLE JAVA.

ALLOW PARALLEL o DISALLOW PARALLEL

Esta cláusula opcional especifica si, para una referencia individual a la función, puede paralelizarse la invocación de la función. En general, las invocaciones de la mayoría de funciones escalares deben poder paralelizarse, pero pueden haber funciones (por ejemplo, las que dependen de una sola copia de una memoria de trabajo) que no puedan. Si se especifica ALLOW PARALLEL o DISALLOW PARALLEL para una función escalar, entonces DB2 aceptará esta especificación. Deben tenerse en cuenta las siguientes cuestiones al determinar qué palabra clave es la adecuada para la función.

- ¿Son todas las invocaciones de la UDF completamente independientes las unas de las otras? En caso afirmativo, especifique ALLOW PARALLEL.
- ¿Se actualiza la memoria de trabajo con cada invocación de la UDF, proporcionando valores que son de interés para la siguiente invocación? (Por ejemplo, el incremento de un contador.) En caso afirmativo, especifique DISALLOW PARALLEL o acepte el valor por omisión.
- ¿Realiza la UDF alguna acción externa que sólo deba producirse en una sola partición? En caso afirmativo, especifique DISALLOW PARALLEL o acepte el valor por omisión.

CREATE FUNCTION (Escalar externa)

- ¿Se utiliza la memoria de trabajo, pero requiere realizar algún proceso de inicialización costoso un número mínimo de veces? En caso afirmativo, especifique ALLOW PARALLEL.

En cualquier caso, el cuerpo de cada función externa debe estar en un directorio que sea accesible en todas las particiones de la base de datos.

El valor por omisión es ALLOW PARALLEL, excepto si se especifica una o varias de las opciones siguientes en la sentencia.

- NOT DETERMINISTIC
- EXTERNAL ACTION
- SCRATCHPAD
- FINAL CALL

Si se especifica o está implícita alguna de estas opciones, el valor por omisión es DISALLOW PARALLEL.

NO DBINFO o DBINFO

Esta cláusula opcional especifica si se pasará cierta información específica conocida por DB2 a la UDF como un argumento en tiempo de una invocación adicional (DBINFO) o no (NO DBINFO). NO DBINFO es el valor por omisión. DBINFO no está soportada para LANGUAGE OLE (SQLSTATE 42613) ni para PARAMETER STYLE JAVA.

Si se especifica DBINFO, entonces se pasa una estructura a la UDF que contiene la información siguiente:

- Nombre de base de datos - el nombre de la base de datos conectada actualmente.
- ID de aplicación - ID de aplicación exclusivo que se establece para cada conexión con la base de datos.
- ID de autorización de aplicación - el ID de autorización de ejecución de la aplicación, independientemente de las UDF anidadas existentes entre esta UDF y la aplicación.
- Página de códigos - identifica la página de códigos de la base de datos.
- Nombre de esquema - exactamente bajo las mismas condiciones que para el Nombre de tabla, contiene el nombre del esquema; de lo contrario, está en blanco.
- Nombre de tabla - sólo si la referencia a UDF está en el lado derecho de una cláusula SET en una sentencia UPDATE o un elemento de la lista VALUES de una sentencia INSERT, contiene el nombre no calificado de la tabla que se está actualizando o insertando; de lo contrario, está en blanco.
- Nombre de columna - exactamente bajo las mismas condiciones que para el Nombre de tabla, contiene el nombre de la columna que se está actualizando o insertando; de lo contrario está en blanco.

CREATE FUNCTION (Escalar externa)

- Versión/release de base de datos - identifica la versión, release y nivel de modificación del servidor de bases de datos que invoca la UDF.
- Plataforma - contiene el tipo de plataforma del servidor.
- Números de columna del resultado de función de la tabla - no es aplicable a las funciones escalares externas.

Consulte el manual *Application Development Guide* para obtener información detallada sobre la estructura y cómo se pasa a la función definida por el usuario.

TRANSFORM GROUP *nombre-grupo*

Indica el grupo de transformación que se debe utilizar, cuando se invoca la función, para las transformaciones de tipo estructurado definidas por el usuario. Es necesaria una transformación si la definición de la función incluye un tipo estructurado definido por el usuario, ya sea como parámetro o como tipo de datos. Si no se especifica esta cláusula, se utiliza el nombre de grupo por omisión, DB2_FUNCTION. Si el *nombre-grupo* especificado (o tomado por omisión) no está definido para un tipo estructurado referenciado, se produce en error (SQLSTATE 42741). Si una función de transformación necesaria FROM SQL o TO SQL no está definida para el nombre de grupo y tipo estructurado proporcionados, se produce un error (SQLSTATE 42744).

Las funciones de transformación, FROM SQL y TO SQL, ya sea especificadas explícita o implícitamente, deben ser funciones SQL que realicen una transformación adecuada entre el tipo estructurado y sus atributos de tipo incorporados.

PREDICATES

Define el filtrado y/o la utilización de la extensión de índice que se lleva a cabo cuando la función se utiliza en un predicado. Una especificación de predicado permite especificar la cláusula opcional SELECTIVITY de una condición de búsqueda. Si se especifica la cláusula PREDICATES, la función debe definirse como DETERMINISTIC con NO EXTERNAL ACTION (SQLSTATE 42613).

WHEN *operador-comparación*

Determina un uso específico de la función en un predicado mediante un operador de comparación ("=", "<", ">", ">=", "<=", "<>").

constante

Especifica un valor constante con un tipo de datos comparable con el tipo RETURNS de la función (SQLSTATE 42818). Cuando un predicado utiliza esta función con el mismo operador de comparación y esta constante, el optimizador puede utilizar el filtrado y la explotación del índice.

EXPRESSION AS *nombre-expresión*

Proporciona un nombre para una expresión. Cuando un predicado utiliza esta función con el mismo operador de comparación y una expresión, puede utilizarse el filtrado y la explotación del índice. La expresión se asigna como nombre de expresión, para que pueda utilizarse como argumento de una función de búsqueda. El *nombre-expresión* no puede ser igual a ningún *nombre-parámetro* de la función que se está creando (SQLSTATE 42711). Cuando se especifica una expresión, se identifica el tipo de la expresión.

FILTER USING

Permite especificar la utilización de una función externa o expresión CASE para un mayor filtrado de la tabla resultante.

invocación-función

Especifica una función de filtro que se puede utilizar para realizar un filtrado adicional de la tabla resultante. Esto es una versión de la función definida (utilizada en el predicado) que reduce el número de filas en el predicado definido por el usuario, para determinar si las filas cumplen los requisitos. Si los resultados producidos por el índice son cercanos a los resultados previstos para el predicado definido por el usuario, puede no ser necesario aplicar la función de filtrado. Si no se especifica esta opción, no se realiza el filtrado de datos.

Esta función puede utilizar como argumento cualquier *nombre-parámetro*, el *nombre-expresión* o constantes (SQLSTATE 42703), y devuelve un entero (SQLSTATE 428E4). Si se devuelve el valor 1, significa que se conserva la fila; en otro caso se descarta.

Esta función también debe cumplir estas condiciones:

- no estar definida con LANGUAGE SQL (SQLSTATE 429B4)
- no estar definida con NOT DETERMINISTIC ni EXTERNAL ACTION (SQLSTATE 42845)
- no tener un tipo de datos estructurado para ninguno de los parámetros (SQLSTATE 428E3)
- no incluir una subconsulta (SQLSTATE 428E4).

Si un argumento invoca otra función u otro método, estas cuatro reglas también se aplican para la función o el método anidado. Sin embargo, los métodos de observador generados por el sistema pueden utilizarse como argumentos de la función de filtro (o de cualquier función o método utilizado como argumento), siempre que el resultado de evaluar el argumento sea un tipo de datos incorporado.

expresión-case

Especifica una expresión CASE para realizar un mayor filtrado de

CREATE FUNCTION (Escalar externa)

la tabla resultante. La *cláusula-when-buscada* y la *cláusula-when-simple* pueden utilizar *nombre-parámetro*, *nombre-expresión* o una constante (SQLSTATE 42703). Como expresión-resultado se puede utilizar una función externa con las reglas especificadas en FILTER USING *invocación-función*. Cualquier función o método referenciado en la expresión-case debe también seguir las cuatro reglas descritas para *invocación-función*.

No se pueden utilizar subconsultas en ningún lugar de la *expresión-case* (SQLSTATE 428E4).

La expresión case debe devolver un valor entero (SQLSTATE 428E4). Si se devuelve el valor 1, significa que se conserva la fila; en otro caso se descarta.

explotación-índice

Define un conjunto de reglas para el método de búsqueda de una extensión de índice que se puede utilizar para explotar el índice.

SEARCH BY INDEX EXTENSION *nombre-extensión-índice*

Identifica la extensión de índice. El *nombre-extensión-índice* debe identificar una extensión de índice existente.

EXACT

Indica que la búsqueda por índice es exacta respecto a la evaluación del predicado. Utilice EXACT para indicar a DB2 que no es necesario aplicar la función de predicado original definida por el usuario ni el filtro después la búsqueda por índice. El predicado definido como EXACT es útil cuando la búsqueda por índice devuelve los mismos resultados que el predicado.

Si no se especifica EXACT, después de la búsqueda por índice se aplica el predicado original definido por el usuario. Si se prevé que el índice sólo proporcione una aproximación del predicado, no especifique la opción EXACT.

Si no se utiliza la búsqueda por índice, es necesario aplicar la función de filtro y el predicado original.

regla-explotación

Describe los patrones y argumentos de la búsqueda y cómo se pueden utilizar para realizar una búsqueda por índice mediante un método de búsqueda definido en la extensión de índice.

WHEN KEY (*nombre-parámetro1*)

Define el patrón de búsqueda. Se puede especificar un solo patrón de búsqueda para una clave. El valor *nombre-parámetro1* identifica los nombres de parámetros de la función definida (SQLSTATE 42703 ó 428E8).

El tipo de datos de *nombre-parámetro1* debe coincidir con el de la clave de origen especificada en la extensión de índice (SQLSTATE 428EY). La coincidencia debe ser exacta para los tipos de datos incorporados y diferenciados y debe darse dentro de la misma jerarquía de tipos en el caso de tipos estructurados.

El resultado de esta cláusula es verdadero cuando los valores del parámetro indicado son columnas que están abarcadas por un índice, de acuerdo con la extensión de índice especificada.

USE *nombre-método-búsqueda*(*nombre-parámetro2*,...)

Define el argumento de búsqueda. Identifica qué método de búsqueda utilizar de entre los definidos en la extensión de índice. El *nombre-método-búsqueda* debe coincidir con un método de búsqueda definido en la extensión de índice (SQLSTATE 42743). Los valores de *nombre-parámetro2* identifican nombres de parámetros de la función definida o el *nombre-expresión* de la cláusula EXPRESSION AS (SQLSTATE 42703). Debe ser diferente de cualquier nombre de parámetro especificado en el patrón de búsqueda (SQLSTATE 428E9). El número de parámetros y el tipo de datos de cada *nombre-parámetro2* debe coincidir con los parámetros definidos para el método de búsqueda en la extensión de índice (SQLSTATE 42816). La coincidencia debe ser exacta para los tipos de datos incorporados y diferenciados, y darse dentro de la misma jerarquía de tipos en el caso de tipos estructurados.

Notas

- La determinación de si un tipo de datos es convertible a otro no tiene en cuenta la longitud, precisión ni escala de los tipos de datos parametrizados, como son CHAR y DECIMAL. Por esta razón, pueden producirse errores al utilizar una función como resultado del intento de conversión de un valor del tipo de datos fuente a un valor del tipo de datos de destino. Por ejemplo, VARCHAR es convertible a DATE, pero si el tipo fuente está realmente definido como VARCHAR(5), al utilizar la función se producirá un error.
- Al elegir los tipos de datos para los parámetros de una función definida por el usuario, tenga en cuenta las reglas para la promoción que afectarán a los valores de entrada (consulte el apartado “Promoción de los tipos de datos” en la página 100). Por ejemplo, una constante que puede utilizarse como un valor de entrada puede tener un tipo de datos incorporado diferente al que se espera y, lo que es más significativo, es posible que no se promueva al tipo de datos que se espera. De acuerdo con las reglas para la promoción, suele aconsejarse la utilización de los siguientes tipos de datos para parámetros:
 - INTEGER en lugar de SMALLINT
 - DOUBLE en lugar de REAL

CREATE FUNCTION (Escalar externa)

- VARCHAR en lugar de CHAR
- VARGRAPHIC en lugar de GRAPHIC
- Para la portabilidad de las UDF a través de plataformas, no deben utilizarse los siguientes tipos de datos:
 - FLOAT- utilice DOUBLE o REAL en su lugar.
 - NUMERIC- utilice DECIMAL en su lugar.
 - LONG VARCHAR- utilice CLOB (o BLOB) en su lugar.
- Una función y un método no pueden prevalecer el uno sobre el otro (SQLSTATE 42745). Para obtener más información sobre la invalidación de un valor por otro, vea “CREATE TYPE (Estructurado)” en la página 923.
- Una función no puede tener la misma signatura que un método (cuando se compara el primer *tipo-parámetro* de la función con el *tipo-sujeto* del método) (SQLSTATE 42723).
- Para obtener información sobre cómo escribir, compilar y enlazar una función externa definida por el usuario, consulte el manual *Application Development Guide*.
- La creación de una función con un nombre de esquema que aún no existe dará como resultado la creación implícita de dicho esquema siempre que el ID de autorización de la sentencia tenga la autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN en el esquema se otorga a PUBLIC.

Ejemplos

Ejemplo 1: Pellow registra la función CENTRE en su esquema PELLOW. Supongamos que esas palabras clave son las que hay por omisión, y que el sistema proporciona un nombre específico de función:

```
CREATE FUNCTION CENTRE (INT, FLOAT)
  RETURNS FLOAT
  EXTERNAL NAME 'mod!middle'
  LANGUAGE C
  PARAMETER STYLE DB2SQL
  DETERMINISTIC
  NO SQL
  NO EXTERNAL ACTION
```

Ejemplo 2: Ahora, McBride (que tiene la autorización DBADM) registra otra función CENTRE en el esquema PELLOW, dándole un nombre explícito específico para el uso posterior del lenguaje de definición de datos y proporcionando explícitamente todos los valores de palabras clave. Observe que esta función utiliza una memoria de trabajo y que, presumiblemente, está acumulando datos que afectan a los resultados posteriores. Debido a que está especificado DISALLOW PARALLEL, cualquier referencia a la función no se paraleliza y, por lo tanto, se utiliza una sola memoria de trabajo para realizar cierta inicialización una sola vez y guardar los resultados.

CREATE FUNCTION (Escalar externa)

```
CREATE FUNCTION PELLOW.CENTRE (FLOAT, FLOAT, FLOAT)
  RETURNS DECIMAL(8,4) CAST FROM FLOAT
  SPECIFIC FOCUS92
  EXTERNAL NAME 'effects!focalpt'
  LANGUAGE C PARAMETER STYLE DB2SQL
  DETERMINISTIC FENCED NOT NULL CALL NO SQL NO EXTERNAL ACTION
  SCRATCHPAD NO FINAL CALL
  DISALLOW PARALLEL
```

Ejemplo 3: A continuación se muestra el programa de función definida por el usuario en lenguaje C para implantar esta regla:

```
output = 2 * input - 4
```

devolviéndose NULL si y sólo si la entrada es nula. Si la sentencia CREATE FUNCTION hubiera utilizado NOT NULL CALL, aún se podría haber escrito de forma más sencilla (es decir, sin la comprobación de nulos). Puede encontrar más ejemplos de los programas de funciones definidas por el usuario en el manual *Application Development Guide*. La sentencia CREATE FUNCTION:

```
CREATE FUNCTION ntest1 (SMALLINT)
  RETURNS SMALLINT
  EXTERNAL NAME 'ntest1!nudft1'
  LANGUAGE C PARAMETER STYLE DB2SQL
  DETERMINISTIC NOT FENCED NULL CALL
  NO SQL NO EXTERNAL ACTION
```

El código del programa:

```
#include "sqlsystem.h"
/* NUDFT1 ES UNA FUNCIÓN ESCALAR DEFINIDA POR EL USUARIO */
/* udft1 acepta como entrada argumentos smallint
  y produce como salida argumentos smallint
  e implanta la regla:
  if (input is null)
    set output = null;
  else
    set output = 2 * input - 4;
*/
void SQL_API_FN nudft1
(short *input, /* puntero al argumento de entrada */
 short *output, /* puntero al resultado */
 short *input_ind, /* puntero a la variable indicadora de entrada */
 short *output_ind, /* puntero a la variable indicadora de salida */
 char sqlstate[6], /* sqlstate, permite términos nulos */
 char fname[28], /* nombre de función completo, término nulo */
 char finst[19], /* nombre específico de función, término nulo */
 char msgtext[71]) /* alm.intermedio texto mensaje, término nulo */
{
  /* comprobar primero si la entrada es nula */
  if (*input_ind == -1)
  {
    /* si la entrada es nula, hacer nula la salida */
    *output_ind = -1;
  }
}
```

CREATE FUNCTION (Escalar externa)

```
}
else
{
  /* si la entrada no es nula, establecer la salida en 2*input-4 */
  *output = 2 * (*input) - 4;
  /* establecer el indicador de salida nula en cero */
  *output_ind = 0;
}

/* marcar finalización correcta dejando sqlstate sin cambiar */
/* y salir */
return;
}
/* end of UDF: NUDFT1 */
```

Ejemplo 4: Lo siguiente registra una UDF Java que devuelve la posición de la primera volcar de una serie. La UDF está escrita en Java, se ha de ejecutar con restricciones de recursos y es el método findvwl de clase javaUDFs.

```
CREATE FUNCTION findv ( CLOB(100K) )
  RETURNS INTEGER
  FENCED
  LANGUAGE JAVA
  PARAMETER STYLE JAVA
  EXTERNAL NAME 'javaUDFs.findvwl'
  NO EXTERNAL ACTION
  CALLED ON NULL INPUT
  DETERMINISTIC
  NO SQL
```

Ejemplo 5: Este ejemplo describe un predicado definido por el usuario, WITHIN, que utiliza como entrada dos parámetros, g1 y g2, de tipo SHAPE:

```
CREATE FUNCTION within (g1 SHAPE, g2 SHAPE)
  RETURNS INTEGER
  LANGUAGE C
  PARAMETER STYLE DB2SQL
  NOT VARIANT
  NOT FENCED
  NO SQL
  NO EXTERNAL ACTION
  EXTERNAL NAME 'db2sefn!SDESpatilRelations'
  PREDICATES
  WHEN = 1
  FILTER USING mbrOverlap(g1..xmin, g1..ymin, g1..xmax, g1..max,
                          g2..xmin, g2..ymin, g2..xmax, g2..ymax)
  SEARCH BY INDEX EXTENSION gridIndex
  WHEN KEY(g1) USE withinExp1Rule(g2)
  WHEN KEY(g2) USE withinExp1Rule(g1)
```

La descripción de la función WITHIN es similar a la de cualquier función definida por el usuario, pero las adiciones siguientes indican que esta función se puede utilizar en un predicado definido por el usuario.

- **PREDICATES WHEN = 1** indica que cuando esta función aparezca como

```
within(g1, g2) = 1
```

en la cláusula WHERE de una sentencia DML, el predicado debe tratarse como un predicado definido por el usuario y el índice definido por la extensión de índice *gridIndex* se debe utilizar para recuperar las filas que cumplen las condiciones de este predicado. Si se especifica una constante, la constante especificada durante la sentencia DML debe coincidir exactamente con la constante especificada en la sentencia CREATE INDEX. Esta condición se proporciona principalmente para abarcar las expresiones booleanas donde el tipo resultante es un 1 o un 0. En los demás casos, la cláusula EXPRESSION es una elección mejor.

- **FILTER USING mbrOverlap** hace referencia a la función de filtro *mbrOverlap*, que es una versión más sencilla del predicado WITHIN. En el ejemplo anterior, la función *mbrOverlap* utiliza como entrada los rectángulos delimitadores mínimos y determina rápidamente si se solapan o no. Si los rectángulos delimitadores mínimos de las dos figuras de entrada no se solapan, significa que *g1* no está contenido en *g2*. Por tanto, el tuplo se puede descartar sin riesgo, evitando la aplicación del costoso predicado WITHIN.
- La cláusula **SEARCH BY INDEX EXTENSION** indica qué combinaciones de extensión de índice y patrón de búsqueda se puede utilizar para este predicado definido por el usuario.

Ejemplo 6: Este ejemplo describe un predicado definido por el usuario, DISTANCE, que utiliza como entrada dos parámetros, P1 y P2, de tipo POINT:

```
CREATE FUNCTION distance (P1 POINT, P2 POINT)
  RETURNS INTEGER
  LANGUAGE C
  PARAMETER STYLE DB2SQL
  NOT VARIANT
  NOT FENCED
  NO SQL
  NO EXTERNAL ACTION
  EXTERNAL NAME 'db2sefn!SDEDistances'
  PREDICATES
  WHEN > EXPRESSION AS distExpr
  SEARCH BY INDEX EXTENSION gridIndex
  WHEN KEY(P1) USE distanceGrRule(P2, distExpr)
  WHEN KEY(P2) USE distanceGrRule(P1, distExpr)
```

La descripción de la función DISTANCE es similar a la de cualquier función definida por el usuario, pero las adiciones siguientes indican que cuando esta función se utiliza en un predicado, éste es un predicado definido por el usuario.

- **PREDICATES WHEN > EXPRESSION AS distExpr** es otra especificación válida del predicado. Cuando se especifica una expresión en la cláusula

CREATE FUNCTION (Escalar externa)

WHEN, el tipo resultante de esa expresión se utiliza para determinar si el predicado es un predicado definido por el usuario de la sentencia DML. Por ejemplo:

```
SELECT T1.C1
FROM T1, T2
WHERE distance (T1.P1, T2.P1) > T2.C2
```

La especificación de predicado DISTANCE utiliza dos parámetros como entrada y compara los resultados con T2.C2, que es de tipo INTEGER. Debido a que sólo es significativo el tipo de datos de la expresión del lado derecho (a diferencia de cuando se utiliza una constante específica), es mejor elegir la cláusula EXPRESSION en el DDL CREATE FUNCTION para especificar un carácter comodín como valor de comparación.

Como método alternativo, también es válido el siguiente predicado definido por el usuario:

```
SELECT T1.C1
FROM T1, T2
WHERE distance(T1.P1, T2.P1) > distance (T1.P2, T2.P2)
```

Existe actualmente la restricción de que sólo el lado derecho se trata como una expresión; el término en el lado izquierdo es la función definida por el usuario correspondiente al predicado definido por el usuario.

- La cláusula **SEARCH BY INDEX EXTENSION** indica qué combinaciones de extensión de índice y patrón de búsqueda se puede utilizar para este predicado definido por el usuario. En el caso de la función DISTANCE, la expresión designada como distExpr es también uno de los argumentos de búsqueda que se pasa a la función productora de rangos (definida como parte de la extensión de índice). El identificador de expresión se utiliza para definir una nombre para la expresión, que se pasa como argumento a la función productora de rangos.

CREATE FUNCTION (Tabla externa)

Esta sentencia se utiliza para registrar una función escalar externa definida por el usuario en un servidor de aplicaciones.

En la cláusula FROM de una sentencia SELECT, se puede utilizar una *función de tabla* y ésta devuelve una tabla a SELECT de fila en fila.

Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse dinámicamente. Sin embargo, si se aplica la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

Autorización

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Autorización IMPLICIT_SCHEMA para la base de datos, si no existe el nombre de esquema implícito o explícito de la función
- Privilegio CREATEIN para el esquema, si existe el nombre de esquema de la función.

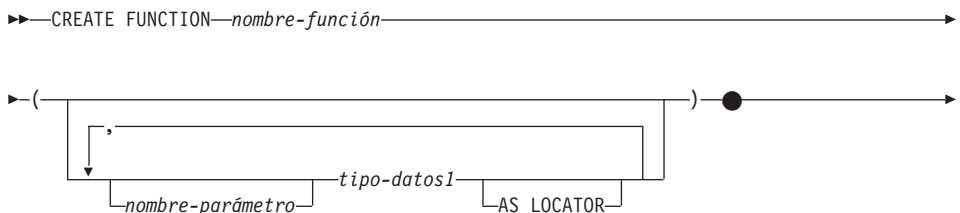
Para crear una función no restringida, el ID de autorización de la sentencia debe también tener como mínimo uno de privilegios siguientes:

- Autorización CREATE_NOT_FENCED para la base de datos
- Autorización SYSADM o DBADM.

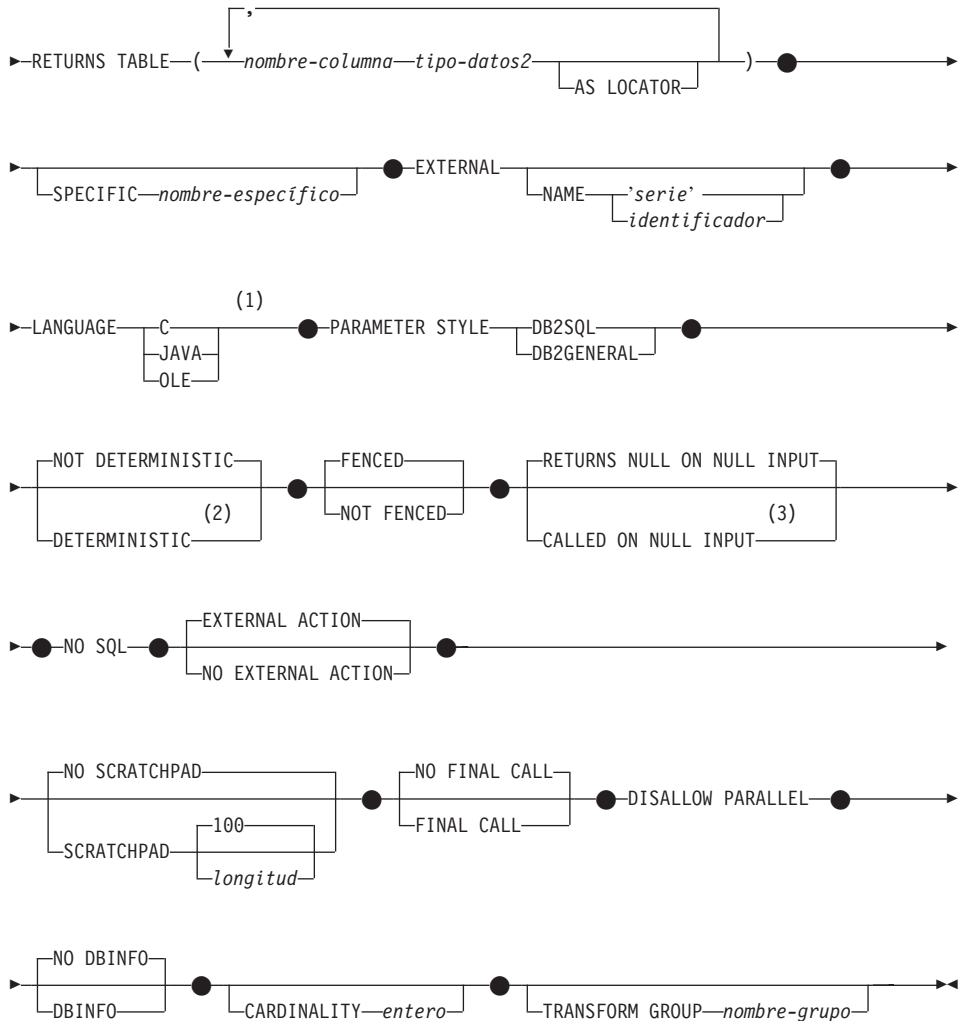
Para crear una función restringida, no se necesitan autorizaciones ni privilegios adicionales.

Si el ID de autorización no tiene la autorización suficiente para realizar la operación, se produce un error (SQLSTATE 42502).

Sintaxis



CREATE FUNCTION (Tabla externa)



Notas:

- 1 Vea “CREATE FUNCTION (Tabla externa OLE DB)” en la página 741 para obtener información sobre cómo crear funciones de tabla externas LANGUAGE OLE DB. Vea “CREATE FUNCTION (SQL, escalar, de tabla o de fila)” en la página 761 para obtener información sobre cómo crear funciones de tabla LANGUAGE SQL.
- 2 Puede especificarse NOT VARIANT en lugar de DETERMINISTIC y VARIANT puede especificarse en lugar de NOT DETERMINISTIC.

- 3 Puede especificarse NULL CALL en lugar de CALLED ON NULL INPUT y puede especificarse NOT NULL CALL en lugar de RETURNS NULL ON NULL INPUT.

Descripción

nombre-función

Indica el nombre de la función que se está definiendo. Consiste en el nombre, calificado o no calificado, que designa una función. La forma no calificada del *nombre-función* es un identificador SQL (cuya longitud máxima es de 18). En las sentencias de SQL dinámico, se utiliza el registro especial CURRENT SCHEMA como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objeto no calificados. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL. El nombre calificado no debe ser el mismo que el tipo de datos del primer parámetro, si ese parámetro es un tipo estructurado.

El nombre, incluidos los calificadores implícitos y explícitos, junto con el número de parámetros y el tipo de datos de cada parámetro (sin tener en cuenta ningún atributo de longitud, precisión o escala del tipo de datos) no debe identificar una función descrita en el catálogo (SQLSTATE 42723). El nombre no calificado, junto con el número y el tipo de datos de los parámetros, que por supuesto es exclusivo en su esquema, no es necesario que lo sea en todos los esquemas.

Si se especifica un nombre compuesto de dos partes, el *nombre-esquema* no puede empezar por "SYS" (SQLSTATE 42939).

Algunos nombres utilizados como palabras clave en predicados están reservados a uso del sistema y no pueden utilizarse como *nombre-función* (SQLSTATE 42939). Estos nombres son SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH y los operadores de comparación que se describen en el apartado "Predicado básico" en la página 213.

Se puede utilizar el mismo nombre para más de una función si hay alguna diferencia en la signatura de las funciones. Aunque no hay ninguna prohibición al respecto, una función de tabla externa definida por el usuario no debe tener el mismo nombre que una función incorporada.

nombre-parámetro

Especifica un nombre opcional para el parámetro que es diferente de los nombres de todos los demás parámetros de la función.

(tipo-datos1,...)

Identifica el número de parámetros de entrada de la función, al tiempo que especifica el tipo de datos de cada parámetro. En la lista debe especificarse una entrada por cada parámetro que la función espera

CREATE FUNCTION (Tabla externa)

recibir. No se permiten más de 90 parámetros. En caso de sobrepasarse este límite, se produce un error (SQLSTATE 54023).

Existe la posibilidad de registrar una función que no tenga ningún parámetro. En este caso, sigue siendo necesario escribir los paréntesis, sin ningún tipo de datos interpuesto. Por ejemplo,

```
CREATE FUNCTION WOOFER() ...
```

En un esquema, no se permite que dos funciones que se denominen igual tengan exactamente el mismo tipo para todos los parámetros correspondientes. Las longitudes, precisiones y escalas no se tienen en cuenta en esta comparación de tipos. Por esta razón, se considera que CHAR(8) y CHAR(35) tienen el mismo tipo, que son DECIMAL(11,2) y DECIMAL(4,3). Hay una agrupación más profunda de los tipos que hace que se traten como el mismo tipo para esta finalidad como, por ejemplo, DECIMAL y NUMERIC. Una signatura duplicada produce un error de SQL (SQLSTATE 42723).

Por ejemplo, si se emiten estas sentencias:

```
CREATE FUNCTION PART (INT, CHAR(15)) ...  
CREATE FUNCTION PART (INTEGER, CHAR(40)) ...  
  
CREATE FUNCTION ANGLE (DECIMAL(12,2)) ...  
CREATE FUNCTION ANGLE (DEC(10,7)) ...
```

la segunda sentencia y la cuarta fallarían porque se considera que son funciones duplicadas.

tipo-datos1

Especifica el tipo de datos del parámetro.

- Pueden proporcionarse especificaciones y abreviaturas de los tipos de datos SQL que pueden especificarse en la definición *tipo-datos* de una sentencia CREATE TABLE y que tienen una correspondencia en el lenguaje que se esté utilizando para escribir la función. Consulte las secciones específicas de los lenguajes del manual *Application Development Guide* para obtener detalles sobre la correlación entre los tipos de datos SQL y los tipos de datos del lenguaje principal con respecto a las funciones definidas por el usuario.
- DECIMAL (y NUMERIC) no son válidos con LANGUAGE C y OLE (SQLSTATE 42815). Para buscar alternativas a la utilización de DECIMAL consulte el manual *Application Development Guide*.
- REF(*nombre-tipo*) puede especificarse como tipo de datos de un parámetro. Sin embargo, dicho parámetro no debe tener ámbito (SQLSTATE 42997).

- Se pueden especificar tipos estructurados, a condición de que existan las funciones de transformación apropiadas en el grupo de transformación asociado.

AS LOCATOR

Para los tipos LOB o los tipos diferenciados que se basan en un tipo LOB, se puede añadir la cláusula AS LOCATOR. Esto indica que se ha de pasar un localizador de LOB a la UDF en lugar del valor real. Esto reduce considerablemente el número de bytes que se pasan a la UDF y puede también mejorar el rendimiento, especialmente cuando la UDF sólo necesite unos pocos bytes. La utilización de los localizadores de LOB de las UDF se describen en el manual *Application Development Guide*.

El siguiente ejemplo ilustra la utilización de la cláusula AS LOCATOR en las definiciones de parámetros:

```
CREATE FUNCTION foo ( CLOB(10M) AS LOCATOR, IMAGE AS LOCATOR)
...
```

lo que supone que IMAGE es un tipo diferenciado basado en uno de los tipos LOB.

Observe también que para fines de promoción de argumentos, la cláusula AS LOCATOR no tiene ningún efecto. En el ejemplo, los tipos se consideran CLOB e IMAGE respectivamente, lo que significa que podría pasarse un argumento CHAR o VARCHAR a la función como el primer argumento. Igualmente, AS LOCATOR no tiene ningún efecto en la signatura de la función, que se utiliza en la comparación de la función (a) cuando se hace referencia en DML, por un proceso llamado "resolución de función" y (b) cuando se hace referencia en una sentencia DDL como, por ejemplo, COMMENT ON o DROP. De hecho, la cláusula puede utilizarse o no en una sentencia COMMENT ON o DROP sin que tenga ningún significado.

Se produce un error (SQLSTATE 42601) si se especifica AS LOCATOR para un tipo que no sea LOB o para un tipo diferenciado basado en un LOB.

Si la función es FENCED, no se puede especificar la cláusula AS LOCATOR (SQLSTATE 42613).

RETURNS TABLE

Especifica que el resultado de la función es una tabla. El paréntesis que sigue a esta palabra clave delimita una lista de nombres y tipos de las columnas de la tabla, parecido al estilo de una sentencia CREATE TABLE

CREATE FUNCTION (Tabla externa)

simple que no tenga especificaciones adicionales (restricciones, por ejemplo). No están permitidas más de 255 columnas (SQLSTATE 54011).

nombre-columna

Especifica el nombre de esta columna. El nombre no puede estar calificado y no se puede utilizar el mismo nombre para más de una columna de la tabla.

tipo-datos2

Especifica el tipo de datos de la columna y puede ser cualquier tipo de datos soportado para un parámetro de una UDF escrita en el lenguaje determinado, excepto para tipos estructurados (SQLSTATE 42997).

AS LOCATOR

Cuando *tipo-datos2* es un tipo LOB o un tipo diferenciado basado en un tipo LOB, la utilización de esta opción indica que la función devuelve un localizador para el valor LOB que tiene su instancia en la tabla resultante.

Los tipos válidos para utilizar en esta cláusula se explican en la página 699.

SPECIFIC *nombre-específico*

Proporciona un nombre exclusivo para la instancia de la función que se está definiendo. Este nombre específico puede utilizarse cuando se utiliza esta función como fuente, al eliminar la función o bien al comentarla. Nunca puede emplearse para invocar a la función. La forma no calificada de *nombre-específico* es un identificador SQL (cuya longitud máxima es 18). La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL. El nombre (incluido el calificador implícito o explícito) no debe identificar a otra instancia de la función que ya exista en el servidor de aplicaciones; de lo contrario, se produce un error. (SQLSTATE 42710).

El *nombre-específico* puede ser el mismo que un *nombre-función* ya existente.

Si no se especifica ningún calificador, se emplea el que se haya utilizado para el *nombre-función*. Si se especifica un calificador, éste debe ser el mismo que el calificador explícito o implícito del *nombre-función*; de lo contrario se produce un error (SQLSTATE 42882).

Si no se especifica el *nombre-específico*, el gestor de bases de datos genera un nombre exclusivo. El nombre exclusivo es SQL seguido de una indicación de fecha y hora en forma de serie caracteres: SQLaammddhhmmssxxx.

EXTERNAL

Esta cláusula indica que la sentencia CREATE FUNCTION se emplea para

CREATE FUNCTION (Tabla externa)

registrar una nueva función basada en el código escrito en un lenguaje de programación externo y cumple los convenios estipulados para los enlaces e interfaces.

Si no se especifica la cláusula NAME se supone "NAME *nombre-función*".

NAME '*serie*'

Esta cláusula identifica al código escrito por el usuario que implanta la función que se está definiendo.

La opción '*serie*' es una constante de serie con un máximo de 254 caracteres. El formato que se utiliza para la serie depende de la opción LANGUAGE especificada.

- Para LANGUAGE C:

La *serie* especificada constituye el nombre de la biblioteca y la función de la biblioteca que el gestor de bases de datos invoca para ejecutar la función definida por el usuario que se está creando (CREATE). Al ejecutar la sentencia CREATE FUNCTION, no es preciso que exista la biblioteca (ni la función dentro de la biblioteca). No obstante, cuando se emplea esta función en una sentencia de SQL, la biblioteca y la función de la biblioteca sí deben existir y estar accesibles desde la máquina que actúa como servidor de bases de datos.

→ ' id_biblioteca [id_vía_absoluta] [!-id_func] ' →

Dentro de los apóstrofes no puede haber ningún espacio en blanco adicional.

id_biblioteca

Identifica el nombre de la biblioteca que contiene la función. El gestor de bases de datos buscará la biblioteca en el directorio `.../sqllib/function` (sistemas basados en UNIX), o en el directorio `... \nombre_instancia\function` (OS/2, y Sistemas operativos Windows de 32 bits según lo especificado por la variable de registro DB2INSTPROF), donde el gestor de bases de datos localizará el directorio sqllib de control que se utiliza para ejecutar el gestor de bases de datos. Por ejemplo, el directorio sqllib de control en sistemas basados en UNIX es `/u/$DB2INSTANCE/sqllib`.

Si '*mifunc*' es el *id_biblioteca* en un sistema basado en UNIX, hace que el gestor de bases de datos busque la función en la biblioteca `/u/production/sqllib/function/mifunc`, siempre que el gestor de bases de datos se ejecute desde `/u/production`.

CREATE FUNCTION (Tabla externa)

En OS/2 y Sistemas operativos Windows de 32 bits, el gestor de bases de datos buscará en LIBPATH o PATH si el *id_biblioteca* no está ubicado en el directorio de la función.

En OS/2, el *id_biblioteca* no debe contener más de 8 caracteres.

id_vía_absoluta

Identifica el nombre completo de vía de acceso de la función.

En un sistema basado en UNIX, por ejemplo, `'/u/jchui/mibib/mifunc'` hace que el gestor de bases de datos busque la función `mifunc` en `/u/jchui/mibib`.

En el OS/2 y Sistemas operativos Windows de 32 bits, `'d:\mibib\mifunc'` hace que el gestor de bases de datos cargue el archivo `mifunc.dll` del directorio `d:\mibib`.

En OS/2, la última parte de esta especificación (es decir, el nombre de la dll), no debe contener más de 8 caracteres.

! id_func

Identifica el nombre del punto de entrada de la función que ha de invocarse. El signo `!` sirve como delimitador entre el ID de biblioteca y el ID de función. Si se omite `! id_func`, el gestor de bases de datos utilizará el valor que había por omisión para el punto de entrada cuando se ha enlazado la biblioteca.

En un sistema basado en UNIX, por ejemplo, `'mimod!func8'` dirige el gestor de bases de datos para que busque la biblioteca `$inst_home_dir/sqllib/function/mimod` y utilice el punto de entrada `func8` dentro de dicha biblioteca.

En el OS/2 y Sistemas operativos Windows de 32 bits, `'mimod!func8'` hace que el gestor de bases de datos cargue el archivo `mimod.dll` y llame a la función `func8()` de la biblioteca de enlace dinámico (DLL).

Si la composición de la serie de caracteres no es correcta, se produce un error (SQLSTATE 42878).

En cualquier caso, el cuerpo de cada función externa debe estar en un directorio que sea accesible en todas las particiones de la base de datos.

- Para LANGUAGE JAVA:

La *serie* especificada contiene el identificador opcional del archivo `jar`, el identificador de clase y el identificador de método, que el gestor de bases de datos invoca para ejecutar la función definida por el usuario que se está creando. No es necesario que existan el identificador de clase ni el identificador de método cuando se

CREATE FUNCTION (Tabla externa)

ejecuta la sentencia CREATE FUNCTION. Si se especifica un *id_jar*, éste debe existir cuando se ejecuta la sentencia CREATE FUNCTION. No obstante, cuando se utiliza la función en una sentencia de SQL, debe existir el identificador de método y debe ser accesible desde la máquina servidora de bases de datos.

→ ' jar_id : id_clase . id_método ' →

Dentro de los apóstrofes no puede haber ningún espacio en blanco adicional.

id_jar

Designa el identificador de jar que se asignó a la colección jar cuando se instaló en la base de datos. Puede ser un identificador simple o un identificador calificado por un esquema. Algunos ejemplos son 'miJar' y 'miEsquema.miJar'

id_clase

Identifica el identificador de clase del objeto Java. Si la clase forma parte de un paquete, la parte del identificador de clase debe incluir el prefijo completo del paquete, por ejemplo 'miPacks.UserFuncs'. La máquina virtual Java buscará en el directorio '.../miPacks/UserFuncs/' correspondiente a las clases. En OS/2 y Sistemas operativos Windows de 32 bits, la máquina virtual Java buscará en el directorio '...\miPacks\UserFuncs\'.

id_método

Identifica el nombre de método del objeto Java que se invoca.

- Para LANGUAGE OLE:

La *serie* especificada es el identificador del programa OLE (*idprog*) o identificador de clase (*idcls*) y el identificador del método, que invoca el gestor de bases de datos para ejecutar la función definida por el usuario que se está creando (CREATE). No es preciso que exista el identificador de programa ni el identificador de clase y el identificador de método al emitir la sentencia CREATE FUNCTION. No obstante, cuando se utiliza la función en una sentencia de SQL, debe existir el identificador de método y debe ser accesible desde la máquina servidora de bases de datos, de lo contrario se produce un error (SQLSTATE 42724).

→ ' idprog | idcls ! id_método ' →

Dentro de los apóstrofes no puede haber ningún espacio en blanco adicional.

CREATE FUNCTION (Tabla externa)

idprog

Identifica el identificador de programa del objeto OLE.

idprog no se interpreta por el gestor de bases de datos sino que sólo se reenvía a la API OLE en tiempo de ejecución. El objeto OLE especificado debe poderse crear y dar soporte a un enlace lógico posterior (denominado también enlace lógico basado en IDispatch).

idcls

Designa el identificador de clase del objeto OLE que se debe crear. Puede utilizarse como una alternativa para especificar *idprog* en el caso de que el objeto OLE no esté registrado con un *idprog*. El *idcls* tiene el formato:

```
{nnnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnnnn}
```

donde 'n' es un carácter alfanumérico. *idcls* no se interpreta por el gestor de bases de datos, sólo se reenvía a las API OLE en el momento de la ejecución.

id_método

Identifica el nombre de método del objeto OLE que se ha de invocar.

NAME *identificador*

Esta cláusula identifica el nombre del código escrito por el usuario que implanta la función que se está definiendo. El *identificador* especificado es un identificador SQL. El identificador SQL se utiliza como el *id-biblioteca* en la serie. A menos que sea un identificador delimitado, se convierte a mayúsculas. Si el identificador está calificado con un nombre de esquema, se ignora la parte correspondiente al nombre de esquema. Este formato de NAME sólo puede utilizarse con LANGUAGE C.

LANGUAGE

Esta cláusula obligatoria se emplea para especificar el convenio de la interfaz de lenguaje en el que se está escrito el cuerpo de la función definida por el usuario.

- C** Esto significa que el gestor de bases de datos llamará a la función definida por el usuario como si se tratase de una función en C. La función definida por el usuario debe ajustarse al convenio de llamadas y enlaces del lenguaje C, tal y como se define en el prototipo de C estándar de ANSI.
- JAVA** Esto significa que el gestor de bases de datos llamará a la función definida por el usuario como un método en una clase Java.
- OLE** Significa que el gestor de bases de datos llamará a la función definida por el usuario como si fuese un método expuesto por un

CREATE FUNCTION (Tabla externa)

objeto de automatización OLE. La función definida por el usuario debe ajustarse a los tipos de datos de automatización OLE y al mecanismo de invocación, tal como se describe en el manual *OLE Automation Programmer's Reference*.

Sólo se da soporte al LANGUAGE OLE para las funciones definidas por el usuario almacenadas en DB2 para Sistemas operativos Windows de 32 bits.

Consulte “CREATE FUNCTION (Tabla externa OLE DB)” en la página 741 para crear las funciones de tabla externa LANGUAGE OLE DB.

PARAMETER STYLE

Esta cláusula se utiliza para especificar los convenios utilizados para pasar parámetros a funciones y devolver el valor de las mismas.

DB2SQL

Se utiliza para especificar los convenios para pasar parámetros a las funciones externas que conforman el lenguaje C de llamada y para devolver los valores de las mismas y para especificar los convenios de enlace. Debe especificarse cuando se utiliza LANGUAGE C o LANGUAGE OLE.

DB2GENERAL

Se utiliza para especificar los convenios para pasar parámetros a funciones externas y para devolver los valores procedentes de esas funciones, las cuales están definidas como método en una clase Java. Sólo se puede especificar cuando se utiliza LANGUAGE JAVA.

El valor DB2GENRL puede utilizarse como sinónimo de DB2GENERAL.

DETERMINISTIC o NOT DETERMINISTIC

Esta cláusula opcional especifica si la función siempre devuelve los mismos resultados para unos valores argumento determinados (DETERMINISTIC) o si la función depende de ciertos valores de estado que afectan a los resultados (NOT DETERMINISTIC). Es decir, una función DETERMINISTIC siempre debe devolver la misma tabla ante invocaciones sucesivas con entradas idénticas. Mediante la especificación de NOT DETERMINISTIC, se evitan las optimizaciones que aprovechan el hecho de que las entradas idénticas siempre produzcan los mismos resultados. Un ejemplo de una función de tabla NOT DETERMINISTIC podría ser una función que recuperase datos de una fuente de datos como, por ejemplo, un archivo.

FENCED o NOT FENCED

Esta cláusula especifica si la función se considera o no “segura” para

CREATE FUNCTION (Tabla externa)

ejecutarse en el proceso o espacio de dirección del entorno operativo del gestor de bases de datos (NOT FENCED) o no (FENCED).

Si una función se registra como FENCED, el gestor de bases de datos aísla los recursos internos de base de datos (por ejemplo, almacenamientos intermedios de datos) para que la función no pueda acceder a ellos. La mayoría de las funciones tienen la opción de ejecutarse como FENCED o NOT FENCED. En general, una función que se ejecute como FENCED no funcionará tan bien como otra de iguales características que se ejecute como NOT FENCED.

Aviso: La utilización de NOT FENCED para funciones no codificadas correctamente puede comprometer la integridad de DB2. DB2 toma algunas precauciones para muchas de las anomalías más habituales que podrían producirse, pero no puede asegurar la integridad completa si se emplean funciones definidas por el usuario como NOT FENCED.

Observe que, aunque el uso de FENCED protege mejor la integridad de la base de datos, una UDF definida como FENCED que no se haya codificado, revisado y probado debidamente puede también causar una anomalía involuntaria de DB2.

Normalmente, la mayoría de las funciones definidas por el usuario se pueden ejecutar como FENCED o NOT FENCED. Una función definida con LANGUAGE OLE sólo admite la especificación FENCED (SQLSTATE 42613).

Para cambiar de FENCED a NOT FENCED, se debe volver a registrar la función (eliminándola y, a continuación, volviendo a crearla). Para que una función definida por el usuario se pueda registrar como NOT FENCED es necesaria la autorización SYSADM, la autorización DBADM o una autorización especial (CREATE_NOT_FENCED).

Si la función es FENCED, no se puede especificar la cláusula AS LOCATOR (SQLSTATE 42613).

RETURNS NULL ON NULL INPUT o CALLED ON NULL INPUT

Esta cláusula opcional puede utilizarse para evitar una llamada a la función externa si cualquiera de los argumentos es nulo. Si la función definida por el usuario está definida para no tener ningún parámetro, entonces por supuesto, esta condición de argumento nulo no puede surgir y no importa cómo se haya codificado esta especificación.

Si se especifica RETURNS NULL ON NULL INPUT y, durante la apertura de la función de tabla, cualquiera de los argumentos de la función es

nulo, no se invoca la función definida por el usuario. El resultado de la función de tabla será una tabla vacía (una tabla sin ninguna fila).

Si se especifica `CALLED ON NULL INPUT`, entonces con independencia de si los argumentos son nulos o no, se invoca la función definida por el usuario. Puede devolver un valor nulo o un valor normal (no nulo). Pero la responsabilidad de comprobar si los valores de argumentos son nulos recae sobre la UDF.

El valor `NULL CALL` puede utilizarse como sinónimo de `CALLED ON NULL INPUT` para mantener la compatibilidad con versiones anteriores. De manera similar, puede utilizarse `NOT NULL CALL` como sinónimo de `RETURNS NULL ON NULL INPUT`.

NO SQL

Esta cláusula obligatoria indica que la función no puede emitir ninguna sentencia de SQL. Si las emite, se produce un error (SQLSTATE 38502) en tiempo de ejecución.

NO EXTERNAL ACTION o EXTERNAL ACTION

Esta cláusula opcional especifica si la función emprende o no alguna acción que cambie el estado de un objeto no gestionado por el gestor de bases de datos. Para evitar las optimizaciones basadas en que la función no produce ningún efecto externo, se especifica `EXTERNAL ACTION`. Por ejemplo: enviar un mensaje, hacer sonar una alarma o grabar un registro en un archivo.

NO SCRATCHPAD o SCRATCHPAD *longitud*

Esta cláusula opcional especifica si debe proporcionarse una memoria de trabajo para una función externa. (Es muy recomendable que las funciones definidas por el usuario sean reentrantes, por lo que una memoria de trabajo proporciona un medio para que la función “guarde el estado” entre una llamada y la siguiente.)

Si se especifica `SCRATCHPAD`, cuando se efectúa la primera invocación de la función definida por el usuario, se asigna memoria para que la función externa utilice una memoria de trabajo. Esta memoria de trabajo tiene las siguientes características:

- *longitud*, si se especifica, define el tamaño en bytes de la memoria de trabajo; este valor debe estar comprendido entre 1 y 32.767 (SQLSTATE 42820). El valor por omisión es 100.
- El valor de inicialización consta sólo de ceros hexadecimales.
- Su ámbito es la sentencia de SQL. Existe una memoria de trabajo para referencia a la función externa en la sentencia de SQL. Por tanto, si la función UDFX de la sentencia siguiente se define con la palabra clave `SCRATCHPAD`, se asignarían dos memorias de trabajo.

CREATE FUNCTION (Tabla externa)

```
SELECT A.C1, B.C2
FROM TABLE (UDFX(:hv1)) AS A,
TABLE (UDFX(:hv1)) AS B
WHERE ...
```

- Su contenido se conserva. Se inicializa al principio de la ejecución de la sentencia y puede ser utilizada por la función de tabla externa para guardar el estado de la memoria de trabajo entre una llamada y la siguiente. Si también se especifica la palabra clave FINAL CALL para la UDF, DB2 no altera NUNCA la memoria de trabajo y, los recursos anclados en ella deben liberarse cuando se emite la llamada especial FINAL.

Si se especifica NO FINAL CALL o se acepta el valor por omisión, la función de tabla externa debe borrar tales recursos en la llamada CLOSE, pues DB2 reinicializará la memoria de trabajo en cada llamada OPEN. Esta elección entre FINAL CALL o NO FINAL CALL y el comportamiento asociado de la memoria de trabajo puede ser un factor importante, especialmente cuando la función de tabla se utiliza en una subconsulta o unión, pues es cuando pueden producirse varias llamadas OPEN durante la ejecución de una sentencia.

- Puede utilizarse como punto central de los recursos del sistema (por ejemplo, la memoria) que podría adquirir la función externa. La función podría adquirir la memoria en la primera llamada, conservar su dirección en la memoria de trabajo y acceder a ella en llamadas posteriores.

(Como se ha indicado anteriormente, la palabra clave FINAL CALL/NO FINAL CALL se utiliza para controlar la reinicialización de la memoria de trabajo y también determina cuándo la función de tabla externa debe liberar los recursos anclados en la memoria de trabajo.)

Si se especifica SCRATCHPAD, en cada invocación de la función definida por el usuario se pasa un argumento adicional a la función externa que indica la dirección de la memoria de trabajo.

Si se especifica NO SCRATCHPAD, no se asignará ni pasará ninguna memoria de trabajo a la función externa.

NO FINAL CALL o FINAL CALL

Esta cláusula opcional especifica si debe realizarse una llamada final (y una primera llamada aparte) a una función externa. También controla cuándo se reinicializa la memoria de trabajo. Si se especifica NO FINAL CALL, DB2 sólo puede realizar tres tipos de llamadas a la función de tabla: open (apertura), fetch (lectura) y close (cierre). En cambio, si se especifica FINAL CALL, además de las llamadas de apertura, lectura y cierre, pueden efectuarse una primera llamada y una llamada final a la función de tabla.

Para las funciones de tabla externa, el argumento tipo-llamada SIEMPRE está presente, sea cual sea la opción que se elija. Vea el manual *Application Development Guide* para obtener más información sobre este argumento y sus valores.

El manual *Application Development Guide* incluye una descripción de la UDF de tabla que procesa estas llamadas cuando se producen errores.

DISALLOW PARALLEL

Esta cláusula especifica si, para una sola referencia a la función, no puede paralelizarse la invocación de la función. Las funciones de tabla siempre se ejecutan en una sola partición.

NO DBINFO o DBINFO

Esta cláusula opcional especifica si se pasará cierta información específica conocida por DB2 a la UDF como un argumento en tiempo de una invocación adicional (DBINFO) o no (NO DBINFO). NO DBINFO es el valor por omisión. DBINFO no está soportada para LANGUAGE OLE (SQLSTATE 42613).

Si se especifica DBINFO, entonces se pasa una estructura a la UDF que contiene la información siguiente:

- Nombre de base de datos - el nombre de la base de datos conectada actualmente.
- ID de aplicación - ID de aplicación exclusivo que se establece para cada conexión con la base de datos.
- ID de autorización de aplicación - el ID de autorización de ejecución de la aplicación, independientemente de las UDF anidadas existentes entre esta UDF y la aplicación.
- Página de códigos - identifica la página de códigos de la base de datos.
- Nombre de esquema - no se puede aplicar a funciones de tabla externas.
- Nombre de tabla - no se puede aplicar a funciones de tabla externas.
- Nombre de columna - no se puede aplicar a funciones de tabla externas.
- Versión/release de base de datos - identifica la versión, release y nivel de modificación del servidor de bases de datos que invoca la UDF.
- Plataforma - contiene el tipo de plataforma del servidor.
- Número de columna del resultado de la función de tabla - una matriz de los números de las columnas del resultado de la función de tabla realmente necesarios para la sentencia en particular que hace referencia a la función. Sólo se proporciona para las funciones de tabla, permite que la UDF optimice devolviendo sólo los valores de columna necesarios en lugar de todos los valores de columna.

CREATE FUNCTION (Tabla externa)

Consulte el manual *Application Development Guide* para ver la información detallada sobre la estructura y cómo se pasa a la UDF.

CARDINALITY *entero*

Esta cláusula opcional proporciona una estimación del número esperado de filas que ha de devolver la función con fines de optimización. Los valores válidos de *entero* están en el rango de 0 a 2.147.483.647 inclusive.

Si no se especifica la cláusula **CARDINALITY** para una función de tabla, DB2 asumirá un valor finito como omisión (el mismo valor asumido para las tablas para las que el programa de utilidad RUNSTATS no ha reunido estadísticas).

Aviso: si una función tiene de hecho una cardinalidad infinita, p. ej. devuelve una fila cada vez que se llama para hacerlo, nunca devuelve una condición de "fin de tabla", las consultas que necesitan una condición de "fin de tabla" para funcionar correctamente serán infinitas y se tendrán que interrumpir. Ejemplos de dichas consultas son los que implican **GROUP BY** y **ORDER BY**. Se aconseja al usuario que no escriba dichas UDF.

TRANSFORM GROUP *nombre-grupo*

Indica el grupo de transformación que se debe utilizar, cuando se invoca la función, para las transformaciones de tipo estructurado definidas por el usuario. Es necesaria una transformación si la definición de la función incluye un tipo estructurado definido por el usuario, en calidad de tipo de datos de parámetro. Si no se especifica esta cláusula, se utiliza el nombre de grupo por omisión, **DB2_FUNCTION**. Si el *nombre-grupo* especificado (o tomado por omisión) no está definido para un tipo estructurado referenciado, se produce en error (SQLSTATE 42741). Si una función de transformación necesaria **FROM SQL** no está definida para el nombre de grupo y tipo estructurado proporcionados, se produce un error (SQLSTATE 42744).

Notas

- Al elegir los tipos de datos para los parámetros de una función definida por el usuario, tenga en cuenta las reglas para la promoción que afectarán a los valores de entrada (consulte el apartado "Promoción de los tipos de datos" en la página 100). Por ejemplo, una constante que puede utilizarse como un valor de entrada puede tener un tipo de datos incorporado diferente al que se espera y, lo que es más significativo, es posible que no se promueva al tipo de datos que se espera. De acuerdo con las reglas para la promoción, suele aconsejarse la utilización de los siguientes tipos de datos para parámetros:
 - **INTEGER** en lugar de **SMALLINT**
 - **DOUBLE** en lugar de **REAL**
 - **VARCHAR** en lugar de **CHAR**

- VARGRAPHIC en lugar de GRAPHIC
- Para asegurar la portabilidad de las UDF de una plataforma a otra, no deben utilizarse los tipos de datos siguientes:
 - FLOAT- utilice DOUBLE o REAL en su lugar.
 - NUMERIC- utilice DECIMAL en su lugar.
 - LONG VARCHAR- utilice CLOB (o BLOB) en su lugar.
- Para obtener información sobre cómo escribir, compilar y enlazar una función externa definida por el usuario, consulte el manual *Application Development Guide*.
- La creación de una función con un nombre de esquema que aún no existe dará como resultado la creación implícita de dicho esquema siempre que el ID de autorización de la sentencia tenga la autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN para el esquema se otorga a PUBLIC.

Ejemplos

Ejemplo 1: Lo siguiente registra una función de tabla escrita para devolver una fila que consta de una columna identificadora de un solo documento para cada documento conocido en un sistema de gestión de texto. El primer parámetro coincide con un área de temas determinada y el segundo parámetro contiene una serie determinada.

Dentro del contexto de una sola sesión, la UDF siempre devolverá la misma tabla y, por lo tanto, se define como DETERMINISTIC. Observe que la cláusula RETURNS que define la salida de DOCMATCH. Debe especificarse FINAL CALL para cada función de tabla. Además, se añade la palabra clave DISALLOW PARALLEL porque las funciones de tabla no pueden funcionar en paralelo. Aunque el tamaño de la salida para DOCMATCH es muy variable, el valor CARDINALITY 20 es representativo y se especifica para ayudar al optimizador DB2.

```
CREATE FUNCTION DOCMATCH (VARCHAR(30), VARCHAR(255))
  RETURNS TABLE (DOC_ID CHAR(16))
  EXTERNAL NAME '/common/docfuncs/rajiv/udfmatch'
  LANGUAGE C
  PARAMETER STYLE DB2SQL
  NO SQL
  DETERMINISTIC
  NO EXTERNAL ACTION
  NOT FENCED
  SCRATCHPAD
  FINAL CALL
  DISALLOW PARALLEL
  CARDINALITY 20
```

Ejemplo 2: Lo siguiente registra una función de tabla OLE que se utiliza para recuperar información de cabecera de mensajes y el texto parcial de los

CREATE FUNCTION (Tabla externa)

mensajes en Microsoft Exchange. Para ver un ejemplo del código que implanta esta función de tabla, consulte el manual *Application Development Guide*.

```
CREATE FUNCTION MAIL()  
    RETURNS TABLE (TIMERECEIVED DATE,  
SUBJECT VARCHAR(15),  
SIZE INTEGER,  
TEXT VARCHAR(30))  
    EXTERNAL NAME 'tfmail.header!list'  
    LANGUAGE OLE  
    PARAMETER STYLE DB2SQL  
    NOT DETERMINISTIC  
    FENCED  
    CALLED ON NULL INPUT  
    SCRATCHPAD  
    FINAL CALL  
    NO SQL  
    EXTERNAL ACTION  
    DISALLOW PARALLEL
```

CREATE FUNCTION (Tabla externa OLE DB)

Esta sentencia se utiliza para registrar una función de tabla externa OLE DB para acceder a los datos de un proveedor OLE DB.

Puede utilizarse una *función de tabla* en la cláusula FROM de SELECT.

Invocación

Esta sentencia se puede intercalar en un programa de aplicación o se puede emitir utilizando sentencias de SQL dinámicas. Es una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

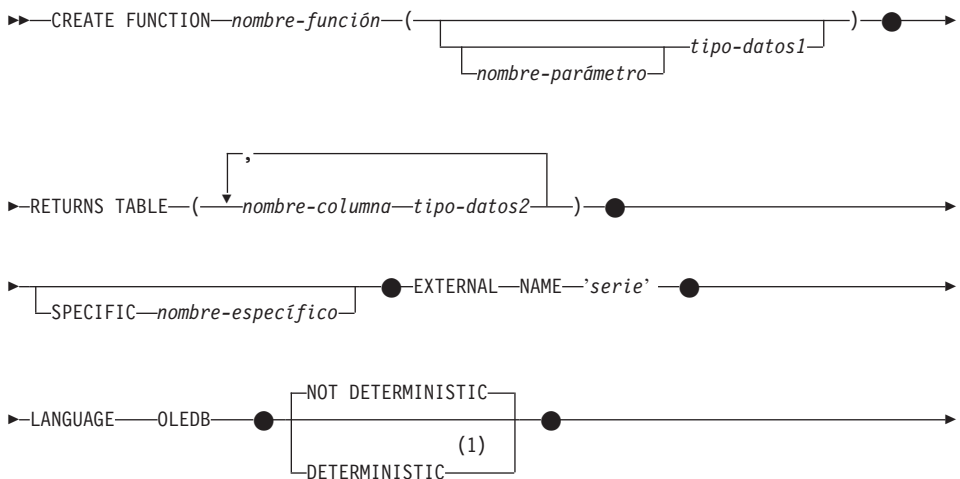
Autorización

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

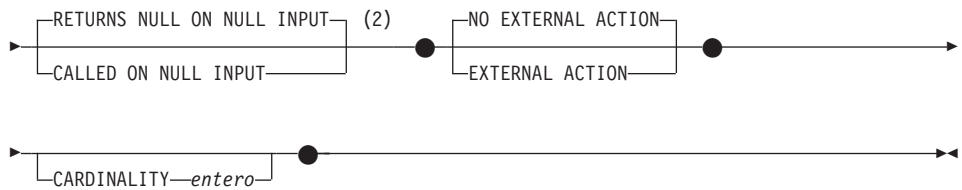
- Autorización SYSADM o DBADM
- Autorización IMPLICIT_SCHEMA para la base de datos, si no existe el nombre de esquema implícito o explícito de la función
- Privilegio CREATEIN para el esquema, si existe el nombre de esquema de la función.

Si el ID de autorización no tiene la autorización suficiente para realizar la operación, se produce un error (SQLSTATE 42502).

Sintaxis



CREATE FUNCTION (Tabla externa OLE DB)



Notas:

- 1 Puede especificarse NOT VARIANT en lugar de DETERMINISTIC y VARIANT puede especificarse en lugar de NOT DETERMINISTIC.
- 2 Puede especificarse NULL CALL en lugar de CALLED ON NULL INPUT y puede especificarse NOT NULL CALL en lugar de RETURNS NULL ON NULL INPUT.

Descripción

nombre-función

Indica el nombre de la función que se está definiendo. Consiste en el nombre, calificado o no calificado, que designa una función. La forma no calificada del *nombre-función* es un identificador SQL (cuya longitud máxima es de 18). En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objeto no calificados. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL.

El nombre, incluidos los calificadores implícitos y explícitos, junto con el número de parámetros y el tipo de datos de cada parámetro (sin tener en cuenta ningún atributo de longitud, precisión o escala del tipo de datos) no debe identificar una función descrita en el catálogo (SQLSTATE 42723). El nombre no calificado, junto con el número y el tipo de datos de los parámetros, que por supuesto es exclusivo en su esquema, no es necesario que lo sea en todos los esquemas.

Si se especifica un nombre compuesto de dos partes, el *nombre-esquema* no puede empezar por "SYS" (SQLSTATE 42939).

Algunos nombres utilizados como palabras clave en predicados están reservados para su uso por el sistema, y no pueden utilizarse como *nombre-función* (SQLSTATE 42939). Los nombres son SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH y los operadores de comparación tal como se describen en "Predicado básico" en la página 213.

Se puede utilizar el mismo nombre para más de una función si hay alguna diferencia en la signatura de las funciones. Aunque no hay

CREATE FUNCTION (Tabla externa OLE DB)

ninguna prohibición al respecto, una función de tabla externa definida por el usuario no debe tener el mismo nombre que una función incorporada.

nombre-parámetro

Especifica un nombre opcional para el parámetro.

tipo-datos1

Identifica el parámetro de entrada de la función y el tipo de datos del parámetro. Si no se especifica ningún parámetro de entrada, los datos se recuperan de la fuente externa posiblemente subestablecida a través de la optimización de consulta. El parámetro de entrada puede ser cualquier tipo de datos de caracteres o de serie gráfica y pasa el texto del mandato al proveedor OLE DB.

Existe la posibilidad de registrar una función que no tenga ningún parámetro. En tal caso, los paréntesis deben seguir codificados, sin tipos de datos intermedios. Por ejemplo,

```
CREATE FUNCTION WOOFER() ...
```

En un esquema, no se permite que dos funciones que se denominen igual tengan exactamente el mismo tipo para todos los parámetros correspondientes. No se tiene en cuenta la longitud en este tipo de comparación. Por lo tanto, se considera que CHAR(8) y CHAR(35) tienen el mismo tipo. Si hay una signatura duplicada se genera un error de SQL (SQLSTATE 42723).

RETURNS TABLE

Especifica que el resultado de la función es una tabla. El paréntesis que sigue a esta palabra clave delimita una lista de nombres y tipos de las columnas de la tabla, parecido al estilo de una sentencia CREATE TABLE simple que no tenga especificaciones adicionales (restricciones, por ejemplo).

nombre-columna

Especifica el nombre de la columna que debe ser igual que el nombre de columna del conjunto de filas correspondiente. El nombre no puede estar calificado y no se puede utilizar el mismo nombre para más de una columna de la tabla.

tipo-datos2

Especifica el tipo de datos de la columna (consulte las secciones específicas del lenguaje de *Application Development Guide* para ver los detalles de la correlación entre los tipos de datos de SQL y los tipos de datos de OLE DB).

SPECIFIC *nombre-específico*

Proporciona un nombre exclusivo para la instancia de la función que se está definiendo. Este nombre específico puede utilizarse cuando se utiliza esta función como fuente, al eliminar la función o bien al comentarla.

CREATE FUNCTION (Tabla externa OLE DB)

Nunca puede emplearse para invocar a la función. La forma no calificada de *nombre-especifico* es un identificador SQL (cuya longitud máxima es 18). La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL. El nombre (incluido el calificador implícito o explícito) no debe identificar a otra instancia de la función que ya exista en el servidor de aplicaciones; de lo contrario, se genera un error. (SQLSTATE 42710).

El *nombre-especifico* puede ser el mismo que un *nombre-función* ya existente.

Si no se especifica ningún calificador, se emplea el que se haya utilizado para el *nombre-función*. Si se especifica un calificador, debe ser el mismo que el calificador explícito o implícito de *nombre-función*, de lo contrario se genera un error (SQLSTATE 42882).

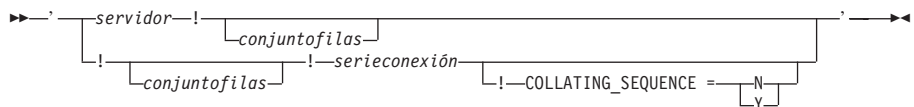
Si no se especifica el *nombre-especifico*, el gestor de bases de datos genera un nombre exclusivo. El nombre exclusivo es SQL seguido de una indicación de fecha y hora en forma de serie caracteres: SQLaammddhhmmssxxx.

EXTERNAL NAME '*serie*'

Esta cláusula identifica la tabla externa y el proveedor OLE DB.

La opción '*serie*' es una constante de serie con un máximo de 254 caracteres.

La serie especificada se utiliza para establecer una conexión y una sesión con un proveedor OLE DB y recuperar los datos de un conjunto de filas. No es necesario que existan el proveedor OLE DB ni la fuente de datos cuando se ejecuta la CREATE FUNCTION. Consulte las funciones de tabla OLE DB de *Application Development Guide* para ver más detalles.



servidor

Identifica el nombre local de una fuente de datos tal como define "CREATE SERVER" en la página 831.

conjuntofilas

Identifica el conjunto de filas (tabla) expuesto por el proveedor OLE DB. Deben proporcionarse nombres de tabla completamente calificados para los proveedores OLE DB que soportan nombres de catálogo o de esquema.

serieconexión

Versión de la serie de las propiedades de inicialización necesaria para conectarse a la fuente de datos. El formato básico de una serie de

CREATE FUNCTION (Tabla externa OLE DB)

conexión se basa en la serie de conexión ODBC. La serie contiene una serie de pares de palabra clave/valor separados por puntos y comas. El signo igual (=) separa cada palabra clave y su valor. Las palabras clave son descripciones de las propiedades de inicialización de OLE DB (conjunto de propiedades DBPROPSET_DBINIT) o palabras clave específicas del proveedor. Consulte las secciones específicas del lenguaje de *Application Development Guide* para ver los detalles.

COLLATING_SEQUENCE

Especifica si la fuente de datos utiliza el mismo orden de clasificación que DB2 Universal Database. Consulte el apartado “CREATE SERVER” en la página 831 para obtener detalles. Los valores válidos son los siguientes:

Y = El mismo orden de clasificación

N = Diferente orden de clasificación

Si no se especifica COLLATING_SEQUENCE, se supone que la fuente de datos tiene un orden de clasificación distinto de DB2 Universal Database.

Si se proporciona *servidor*, *serieconexión* o COLLATING_SEQUENCE no están permitidos en el nombre externo. Se definen como las opciones de servidor CONNECTSTRING y COLLATING_SEQUENCE. Si no se proporciona ningún *servidor*, debe proporcionarse una *serieconexión*. Si no se proporciona *conjuntofilas*, la función de tabla debe tener un parámetro de entrada para realizar el paso a través del texto del mandato al proveedor OLE DB.

LANGUAGE OLEDB

Esto significa que el gestor de bases de datos desplegará un cliente OLE DB genérico incorporado para recuperar los datos de OLE DB. El desarrollado no necesita ninguna implantación de función de tabla.

Las funciones de tabla LANGUAGE OLEDB pueden crearse en cualquier plataforma, pero sólo se ejecutan en plataformas soportadas por Microsoft OLE DB.

DETERMINISTIC o NOT DETERMINISTIC

Esta cláusula opcional especifica si la función siempre devuelve los mismos resultados para unos valores argumento determinados (DETERMINISTIC) o si la función depende de ciertos valores de estado que afectan a los resultados (NOT DETERMINISTIC). Es decir, una función DETERMINISTIC siempre debe devolver la misma tabla ante invocaciones sucesivas con entradas idénticas. Se evitan las optimizaciones que aprovechan el hecho de que entradas idénticas siempre produzcan los mismos resultados especificando NOT DETERMINISTIC.

CREATE FUNCTION (Tabla externa OLE DB)

RETURNS NULL ON NULL INPUT o CALLED ON NULL INPUT

Esta cláusula opcional puede utilizarse para evitar una llamada a la función externa si cualquiera de los argumentos es nulo. Si la función definida por el usuario está definida para no tener parámetros, esta condición de argumento nulo no puede producirse.

Si se especifica RETURNS NULL ON NULL INPUT y si durante la ejecución alguno cualquiera de los argumentos de la función es nulo, no se llama a la función definida por el usuario y el resultado es la tabla vacía, p. ej. una tabla sin filas.

Si se especifica CALLED ON NULL INPUT, entonces durante la ejecución sin tener en cuenta si los argumentos son o no nulos, se invoca la función definida por el usuario. Puede devolver una tabla vacía o no, dependiendo de su lógica. Pero la responsabilidad de comprobar si los valores de argumentos son nulos recae sobre la UDF.

El valor NULL CALL puede utilizarse como sinónimo de CALLED ON NULL INPUT para mantener la compatibilidad con versiones anteriores. De manera similar, puede utilizarse NOT NULL CALL como sinónimo de RETURNS NULL ON NULL INPUT.

NO EXTERNAL ACTION o EXTERNAL ACTION

Esta cláusula opcional especifica si la función realiza alguna acción que cambia el estado de un objeto no gestionado por el gestor de bases de datos. Para evitar las optimizaciones basadas en que la función no produce ningún efecto externo, se especifica EXTERNAL ACTION. Por ejemplo: enviar un mensaje, hacer sonar una alarma o grabar un registro en un archivo.

CARDINALITY *entero*

Esta cláusula opcional proporciona una estimación del número esperado de filas que ha de devolver la función con fines de optimización. Los valores válidos de *entero* están en el rango de 0 a 2 147 483 647 inclusive.

Si no se especifica la cláusula CARDINALITY para una función de tabla, DB2 asumirá un valor finito como omisión (el mismo valor asumido para las tablas para las que el programa de utilidad RUNSTATS no ha reunido estadísticas).

Aviso: si una función tiene de hecho una cardinalidad infinita, p. ej. devuelve una fila cada vez que se llama para hacerlo, nunca devuelve una condición de "fin de tabla", las consultas que necesitan una condición de "fin de tabla" para funcionar correctamente serán infinitas y se tendrán que interrumpir. Ejemplos de dichas consultas son los que implican GROUP BY y ORDER BY. Se aconseja al usuario que no escriba dichas UDF.

Notas

- FENCED, FINAL CALL, SCRATCHPAD, PARAMETER STYLE DB2SQL, DISALLOW PARALLEL, NO DBINFO y NO SQL está implícitos en la sentencia y se pueden especificar. Consulte “CREATE FUNCTION (Tabla externa)” en la página 723 para ver las descripciones específicas.
- Al elegir los tipos de datos para los parámetros de una función definida por el usuario, tenga en cuenta las reglas para la promoción que afectarán a sus valores de entrada (consulte la sección “Promoción de los tipos de datos” en la página 100). Por ejemplo, una constante que puede utilizarse como un valor de entrada puede tener un tipo de datos incorporado diferente al que se espera y, lo que es más significativo, es posible que no se promueva al tipo de datos que se espera. De acuerdo con las reglas para la promoción, suele aconsejarse la utilización de los siguientes tipos de datos para parámetros:
 - VARCHAR en lugar de CHAR
 - VARGRAPHIC en lugar de GRAPHIC
- Para asegurar la portabilidad de las UDF de una plataforma a otra, no deben utilizarse los tipos de datos siguientes:
 - FLOAT- utilice DOUBLE o REAL en su lugar.
 - NUMERIC- utilice DECIMAL en su lugar.
 - LONG VARCHAR- utilice CLOB (o BLOB) en su lugar.
- Para obtener información sobre la creación de una función de tabla externa OLE DB definida por el usuario, consulte *Application Development Guide*.
- La creación de una función con un nombre de esquema que aún no existe dará como resultado la creación implícita de dicho esquema siempre que el ID de autorización de la sentencia tenga la autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN en el esquema se otorga a PUBLIC.

Ejemplos

Ejemplo 1: Lo siguiente registra una función de tabla OLE DB, que recupera información de clasificación de una base de datos Microsoft Access. La serie de conexión se define en el nombre externo.

```
CREATE FUNCTION orders ()
  RETURNS TABLE (orderid INTEGER,
                 customerid CHAR(5),
                 employeeid INTEGER,
                 orderdate TIMESTAMP,
                 requireddate TIMESTAMP,
                 shippeddate TIMESTAMP,
                 shipvia INTEGER,
                 freight dec(19,4))
```

CREATE FUNCTION (Tabla externa OLE DB)

```
LANGUAGE OLEDB
EXTERNAL NAME '!orders!Provider=Microsoft.Jet.OLEDB.3.51;
              Data Source=c:\sql11ib\samples\oledb\nwind.mdb
!COLLATING_SEQUENCE=Y';
```

Ejemplo 2: Lo siguiente registra una función de tabla OLE DB, que recupera información de cliente de una base de datos Oracle. La serie de conexión se proporciona a través de una definición de servidor. El nombre de tabla está completamente calificado en el nombre externo. El usuario local john se correlaciona con el usuario remoto dave. Otros usuarios utilizarán el id de usuario de invitado en la serie de conexión. Consulte “CREATE SERVER” en la página 831, “CREATE WRAPPER” en la página 974 y “CREATE USER MAPPING” en la página 955 para ver los detalles de las sentencias.

```
CREATE SERVER spirit
WRAPPER OLEDB
OPTIONS (CONNECTSTRING 'Provider=MSDAORA;Persist Security Info=False;
                       User ID=guest;password=pwd;Locale Identifier=1033;
                       OLE DB Services=CLIENTCURSOR;Data Source=spirit');

CREATE USER MAPPING FOR john
SERVER spirit
OPTIONS (REMOTE_AUTHID 'dave', REMOTE_PASSWORD 'mypwd');

CREATE FUNCTION customers ()
RETURNS TABLE (customer_id INTEGER,
                name VARCHAR(20),
                address VARCHAR(20),
                city VARCHAR(20),
                state VARCHAR(5),
                zip_code INTEGER)

LANGUAGE OLEDB
EXTERNAL NAME 'spirit!demo.customer';
```

Ejemplo 3: Lo siguiente registra una función de tabla OLE DB, que recupera información sobre tiendas a partir de la base de datos MS SQL Server 7.0. La serie de conexión se proporciona en el nombre externo. La función de tabla tiene un parámetro de entrada para el paso a través del texto del mandato al proveedor OLE DB. No es necesario especificar el nombre del conjunto de filas en el nombre externo. La consulta del ejemplo pasa un mandato SQL para recuperar las 3 tiendas con mayor nivel de ventas.

```
CREATE FUNCTION favorites (varchar(600))
RETURNS TABLE (store_id CHAR (4),
                name VARCHAR (41),
                sales INTEGER)

SPECIFIC favorites
LANGUAGE OLEDB
EXTERNAL NAME '!!Provider=SQLOLEDB.1;Persist Security Info=False;
              User ID=sa;Initial Catalog=pubs;Data Source=WALTZ;
              Locale Identifier=1033;Use Procedure for Prepare=1;
              Auto Translate=False;Packet Size=4096;Workstation ID=WALTZ;
              OLE DB Services=CLIENTCURSOR;';
```

CREATE FUNCTION (Tabla externa OLE DB)

```
SELECT *
  FROM TABLE (favorites
    (' select top 3 sales.stor_id as store_id, ' || '
      stores.stor_name as name, ' || '
      sum(sales.qty) as sales ' || '
    from sales, stores ' || '
    where sales.stor_id = stores.stor_id ' || '
    group by sales.stor_id, stores.stor_name' || '
    order by sum(sales.qty) desc')) as f;
```

CREATE FUNCTION (fuente o plantilla)

CREATE FUNCTION (fuente o plantilla)

Esta sentencia se utiliza para:

- Registrar una función definida por el usuario, basada en otra función escalar o de columna existente, en un servidor de aplicaciones.
- Registrar una plantilla de función en un servidor de aplicaciones que está designado como servidor federado. Una *plantilla de función* es una función parcial que no contiene código ejecutable. El usuario lo crea con la finalidad de correlacionarlo con una función fuente de datos. Después de crear la correlación, el usuario puede especificar la plantilla de función en consultas que se envían al servidor federado. Cuando se procesa una consulta como esta, el servidor federado invocará la función fuente de datos a la que está asignada la plantilla y devolverá los valores cuyos tipos de datos correspondan con los de la opción RETURNS de la definición de la plantilla. Vea “Correlaciones de funciones, plantillas de funciones y opciones de correlación de funciones” en la página 63 para obtener más información.

Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse dinámicamente. Sin embargo, si se aplica la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

Autorización

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Autorización IMPLICIT_SCHEMA para la base de datos, si no existe el nombre de esquema implícito o explícito de la función
- Privilegio CREATEIN para el esquema, si existe el nombre de esquema de la función.

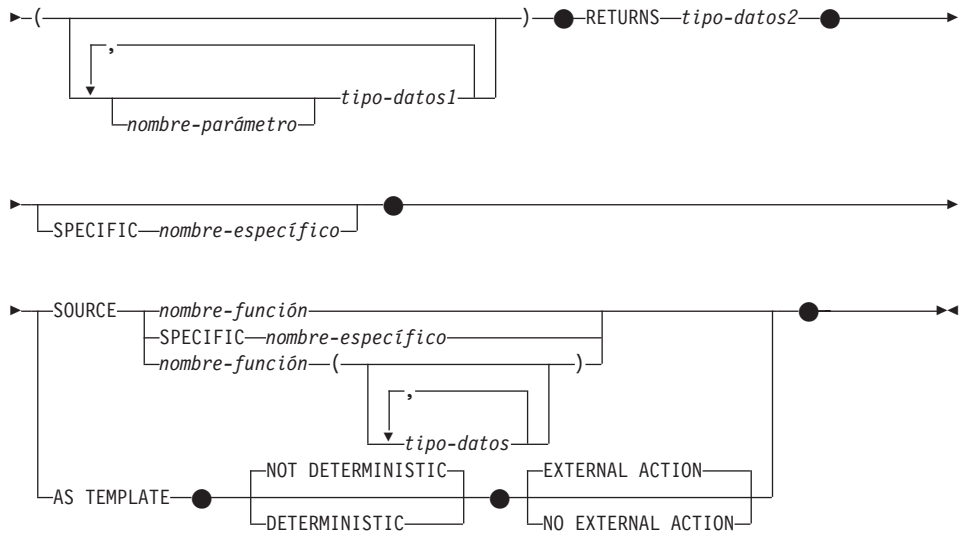
Si el ID de autorización no tiene la autorización suficiente para realizar la operación, se produce un error (SQLSTATE 42502).

No se necesita ninguna autorización para una función referenciada en la cláusula SOURCE.

Sintaxis

►—CREATE FUNCTION—*nombre-función*—►

CREATE FUNCTION (fuente o plantilla)



Descripción

nombre-función

Identifica la función o plantilla de función que se está definiendo. Consiste en el nombre, calificado o no calificado, que designa una función. La forma no calificada del *nombre-función* es un identificador SQL (cuya longitud máxima es de 18). En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace QUALIFIER especifica de forma implícita el calificador para los nombres de objeto no calificados. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL.

El nombre, incluidos los calificadores implícitos y explícitos, junto con el número de parámetros y el tipo de datos de cada parámetro (sin tener en cuenta ningún atributo de longitud, precisión o escala del tipo de datos) no debe identificar una función ni una plantilla de función descritos en el catálogo (SQLSTATE 42723). El nombre no calificado, junto con el número y el tipo de datos de los parámetros, que por supuesto es exclusivo en su esquema, no es necesario que lo sea en todos los esquemas.

Si se especifica un nombre compuesto de dos partes, el *nombre-esquema* no puede empezar por "SYS". De lo contrario, se genera un error (SQLSTATE 42939).

Algunos nombres utilizados como palabras clave en predicados están reservados a uso del sistema y no pueden utilizarse como *nombre-función* (SQLSTATE 42939). Estos nombres son SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR,

CREATE FUNCTION (fuente o plantilla)

MATCH y los operadores de comparación, tal como se describe en el apartado “Predicado básico” en la página 213.

Al nombrar una función definida por el usuario que tenga su fuente en una función ya existente con el fin de dar soporte a la misma función con un tipo diferenciado definido por el usuario, puede utilizarse el mismo nombre que el de la función derivada. De esta forma, los usuarios pueden utilizar la misma función con un tipo diferenciado definido por el usuario sin darse cuenta de que era necesaria una definición adicional. En general, puede utilizarse el mismo nombre para más de una función si existe alguna diferencia en la signatura de las funciones.

(tipo-datos, ...)

Identifica el número de parámetros de entrada de la función o plantilla de función y especifica el tipo de datos de cada parámetro. Debe especificarse una entrada de la lista para cada parámetro que la función o plantilla de función espera recibir. No se permiten más de 90 parámetros. En caso de sobrepasarse este límite, se produce un error (SQLSTATE 54023).

Se puede registrar una función o plantilla de función que no tenga ningún parámetro. En este caso, sigue siendo necesario escribir los paréntesis, sin ningún tipo de datos interpuesto. Por ejemplo,

```
CREATE FUNCTION WOOFER() ...
```

En un esquema, no se permite que dos funciones o plantillas de función con idéntico nombre tengan exactamente el mismo tipo para todos los parámetros correspondientes. (Esta restricción también es aplicable a una función y plantilla de función de un esquema que tengan el mismo nombre). Las longitudes, precisiones y escalas no se tienen en cuenta en esta comparación de tipos. Por esta razón, se considera que CHAR(8) y CHAR(35) tienen el mismo tipo, que son DECIMAL(11,2) y DECIMAL(4,3). Hay una agrupación más profunda de los tipos que hace que se traten como el mismo tipo para esta finalidad como, por ejemplo, DECIMAL y NUMERIC. Una signatura duplicada produce un error de SQL (SQLSTATE 42723).

Por ejemplo, si se emiten estas sentencias:

```
CREATE FUNCTION PART (INT, CHAR(15)) ...  
CREATE FUNCTION PART (INTEGER, CHAR(40)) ...  
  
CREATE FUNCTION ANGLE (DECIMAL(12,2)) ...  
CREATE FUNCTION ANGLE (DEC(10,7)) ...
```

la segunda sentencia y la cuarta fallarían porque se considera que son funciones duplicadas.

CREATE FUNCTION (fuente o plantilla)

nombre-parámetro

Especifica un nombre opcional para el parámetro que es diferente de los nombres de todos los demás parámetros de la función.

tipo-datos1

Especifica el tipo de datos del parámetro.

Con una función escalar derivada se puede utilizar cualquier tipo de datos SQL válido siempre que se pueda convertir al tipo del parámetro correspondiente de la función identificada en la cláusula SOURCE (vea “Conversión entre tipos de datos” en la página 101 para conocer la definición de “convertible”). El tipo de datos REF(*nombre-tipo*) no puede especificarse como tipo de datos de un parámetro (SQLSTATE 42997).

Puesto que la función tiene fuente, no es necesario (aunque sí está permitido) especificar la longitud, precisión o escala para los tipos de datos parametrizados. En su lugar, pueden utilizarse paréntesis (por ejemplo CHAR()). Un *tipo de datos parametrizados* es cualquiera de los tipos de datos que se pueden definir con una longitud, escala o precisión específicas. Los tipos de datos parametrizados son los tipos de datos de serie de caracteres y los tipos de datos decimales.

RETURNS

Esta cláusula obligatoria identifica el resultado de la función o plantilla de función.

tipo-datos2

Especifica el tipo de datos de la salida.

Es válido cualquier tipo de datos SQL, así como un tipo diferenciado, siempre que se pueda convertir desde tipo del resultado de la función fuente (para la definición de “convertible”, consulte el apartado “Conversión entre tipos de datos” en la página 101).

El parámetro de un tipo parametrizado no es necesario especificarlo, tal como se describió anteriormente para los parámetros de una función derivada. En su lugar, se pueden utilizar paréntesis vacíos, como por ejemplo VARCHAR().

En la página 757 hallará una serie de consideraciones y reglas adicionales que se aplican a la especificación del tipo de datos en la cláusula RETURNS cuando la función tiene su fuente en otra.

SPECIFIC *nombre-específico*

Proporciona un nombre exclusivo para la instancia de la función que se está definiendo. Este nombre específico puede utilizarse cuando se utiliza esta función como fuente, al eliminar la función o bien al comentarla. Nunca puede emplearse para invocar a la función. La forma no calificada de *nombre-específico* es un identificador SQL (cuya longitud máxima es 18).

CREATE FUNCTION (fuente o plantilla)

La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL. El nombre, incluido el calificador implícito o explícito, no debe identificar otra instancia de función que exista en el servidor de aplicaciones; de lo contrario, se produce un error. (SQLSTATE 42710).

El *nombre-específico* puede ser el mismo que un *nombre-función* ya existente.

Si no se especifica ningún calificador, se emplea el que se haya utilizado para el *nombre-función*. Si se especifica un calificador, éste debe ser el mismo que el calificador explícito o implícito del *nombre-función*; de lo contrario se produce un error (SQLSTATE 42882).

Si no se especifica el *nombre-específico*, el gestor de bases de datos genera un nombre exclusivo. El nombre exclusivo es SQL seguido de una indicación de fecha y hora en forma de serie caracteres: SQLaammddhhmmssxxx.

SOURCE

Especifica que la función que se esté creando ha de implantarse por otra función (la función fuente), que el gestor de bases de datos ya conoce. La función fuente puede ser una función incorporada⁷² u una función escalar definida por el usuario, creada previamente.

Sólo puede especificarse la cláusula SOURCE para funciones escalares o de columna; no se puede especificar para funciones de tabla.

La cláusula SOURCE proporciona la identidad de la otra función.

nombre-función

Identifica la función determinada que ha de emplearse como fuente y que sólo es válida si en el esquema hay una función específica exactamente con este *nombre-función*. Esta variante de sintaxis no es válida para una función fuente que sea una función incorporada.

Si se proporciona un nombre no calificado, entonces la vía de acceso de SQL actual del ID de autorización (el valor del registro especial CURRENT PATH) se utiliza para localizar la función. Se selecciona el primer esquema de la vía de acceso de la función que tiene una función con este nombre.

Si no hay ninguna función con este nombre en el esquema nombrado o si el nombre no está calificado y no hay ninguna función con este nombre en la vía de acceso de la función, se produce un error (SQLSTATE 42704). Si hay más de una instancia específica de la función en el esquema nombrado o ubicado, se produce un error (SQLSTATE 42725).

72. Con la excepción de COALESCE, NODENUMBER, NULLIF, PARTITION, TYPE_ID, TYPE_NAME, TYPE_SCHEMA y VALUE.

SPECIFIC *nombre-específico*

Identifica una función determinada definida por el usuario que ha de servir como fuente, por el *nombre-específico* ya sea especificado o tomado por omisión en el tiempo de creación de la función. Esta variante de sintaxis no es válida para una función fuente que sea una función incorporada.

Si se proporciona un nombre no calificado, entonces la vía de acceso de SQL actual del ID de autorización se utiliza para localizar la función. Se selecciona el primer esquema de la vía de acceso de la función que tiene una función con este nombre específico.

Si no existe ninguna función con este *nombre-específico* en el esquema nombrado o bien si el nombre no está calificado y no hay ninguna función con este *nombre-específico* en la vía de acceso de SQL, se produce un error (SQLSTATE 42704).

nombre-función (tipo-datos,...)

Proporciona la signatura de la función, que identifica de manera exclusiva la función fuente. Ésta es la única variante de sintaxis válida para una función fuente que es una función incorporada.

Las reglas para la resolución de funciones (descritas en el apartado “Resolución de funciones” en la página 162) se aplican para seleccionar una función entre aquellas funciones que tienen el mismo nombre, dado los tipos de datos que se han especificado en la cláusula SOURCE. Sin embargo, el tipo de datos de cada parámetro de la función seleccionada debe tener exactamente el tipo de datos correspondiente que se ha especificado en la función fuente.

nombre-función

Constituye el nombre de la función fuente. Si se proporciona un nombre no calificado, entonces se tienen en cuenta los esquemas de la vía de acceso de SQL del usuario.

tipo-datos

Debe coincidir con el tipo de datos que se haya especificado en la sentencia CREATE FUNCTION en la posición correspondiente (separado por comas).

Para los tipos de datos parametrizados no es necesario especificar la longitud, la precisión ni la escala. En su lugar, puede haber un conjunto vacío de paréntesis codificados para indicar que esos atributos deben ignorarse al buscar una coincidencia del tipo de datos. Por ejemplo, DECIMAL() coincidirá con un parámetro cuyo tipo de datos esté definido como DECIMAL(7,2)).

No puede utilizarse FLOAT() (SQLSTATE 42601) puesto que el valor del parámetro indica tipos de datos diferentes (REAL o DOUBLE).

CREATE FUNCTION (fuente o plantilla)

De todas formas, si la longitud, la precisión o la escala están codificadas, el valor debe coincidir exactamente con el que se especifica en la sentencia CREATE FUNCTION. Esto es útil para asegurarse de que se va a utilizar exactamente la función que se necesita. Tenga en cuenta que los sinónimos de los tipos de datos también se considerarán coincidencias (por ejemplo, DEC y NUMERIC coincidirán).

No es necesario que un tipo de FLOAT(n) coincida con el valor definido para n puesto que $0 < n < 25$ significa REAL y $24 < n < 54$ significa DOUBLE. La coincidencia se produce basándose en si el tipo es REAL o DOUBLE.

Si en el esquema nombrado o implícito no hay ninguna función con la signatura especificada, se genera un error (SQLSTATE 42883).

AS TEMPLATE

Indica que esta sentencia se utilizará para crear una plantilla de función y no una función con código ejecutable.

DETERMINISTIC o NOT DETERMINISTIC

Esta cláusula opcional especifica si la función siempre devuelve los mismos resultados para unos valores argumento determinados (DETERMINISTIC) o si la función depende de ciertos valores de estado que afectan a los resultados (NOT DETERMINISTIC). Es decir, una función DETERMINISTIC siempre debe devolver la misma tabla ante invocaciones sucesivas con entradas idénticas. Mediante la especificación de NOT DETERMINISTIC, se evitan las optimizaciones que aprovechan el hecho de que las entradas idénticas siempre produzcan los mismos resultados.

Se debe especificar NOT DETERMINISTIC, de forma explícita o implícita, si el cuerpo de la función accede a un registro especial, o invoca otra función no determinista (SQLSTATE 428C2).

NO EXTERNAL ACTION o EXTERNAL ACTION

Esta cláusula opcional especifica si la función emprende o no alguna acción que cambie el estado de un objeto no gestionado por el gestor de bases de datos. Si se especifica NO EXTERNAL ACTION, el sistema puede utilizar determinadas optimizaciones basadas en que la función no produce ningún efecto externo.

Se debe especificar EXTERNAL ACTION, de forma explícita o implícita, si el cuerpo de la función invoca otra función que tiene una acción externa (SQLSTATE 428C2).

Normas

- En esta sección llamaremos CF a la función que se está creando y FS a la función identificada en la cláusula SOURCE, no importa cual de las tres sintaxis permitidas se haya utilizado para identificar FS.
 - El nombre no calificado de la función CF y el nombre no calificado de la SF pueden ser distintos.
 - Una función nombrada como la fuente de otra función puede, a su vez, utilizar otra función como su fuente. Al hacer uso de este recurso deben extremarse las precauciones, puesto que podría resultar muy difícil depurar una aplicación si una función invocada indirectamente genera un error.
 - Si se especifican con la cláusula SOURCE, ninguna de las cláusulas siguientes es válida (porque la CF heredará estos atributos de la SF):
 - CAST FROM ...,
 - EXTERNAL ...,
 - LANGUAGE ...,
 - PARAMETER STYLE ...,
 - DETERMINISTIC / NOT DETERMINISTIC,
 - FENCED / NOT FENCED,
 - RETURNS NULL ON NULL INPUT / CALLED ON NULL INPUT
 - EXTERNAL ACTION / NO EXTERNAL ACTION
 - NO SQL
 - SCRATCHPAD / NO SCRATCHPAD
 - FINAL CALL / NO FINAL CALL
 - RETURNS TABLE (...)
 - CARDINALITY ...
 - ALLOW PARALLEL / DISALLOW PARALLEL
 - DBINFO / NO DBINFO

Si se incumplen estas reglas se produce un error (SQLSTATE 42613).

- El número de parámetros de entrada de la CF debe ser el mismo que los de la SF; de lo contrario, se produce un error (SQLSTATE 42624).
- La CF no tendrá que especificar necesariamente la longitud, precisión ni escala para un tipo de datos parametrizados en los casos siguientes:
 - Los parámetros de entrada de la función.
 - Su parámetro RETURNS

En su lugar, se pueden especificar parámetros vacíos como parte del tipo de datos (por ejemplo: VARCHAR()), para mostrar que la

CREATE FUNCTION (fuente o plantilla)

longitud/precisión/escala serán las mismas que las de la función fuente o las determinadas por la conversión del tipo de datos.

No obstante, si se especifica la longitud, la precisión o la escala, el valor de la CF se comprueba comparándolo con el valor correspondiente de la SF, tal y como se explica más abajo para los parámetros de entrada, y se devuelve un valor.

- La especificación de los parámetros de entrada de la CF se comparan con los de la SF. El tipo de datos de cada parámetro de CF debe ser el mismo que el tipo de datos del parámetro correspondiente de la SF o debe ser *convertible* a él. Hallará una definición del término 'convertible' en el apartado "Conversión entre tipos de datos" en la página 101. Si algún parámetro no es del mismo tipo o no es convertible, se produce un error (SQLSTATE 42879).

Tenga en cuenta que esta regla no proporciona ninguna garantía frente a los errores producidos al utilizar la CF. Un argumento que coincide con el tipo de datos y los atributos de longitud o de precisión de un parámetro de la CF, tal vez no sea asignable si el parámetro correspondiente de la SF tiene una longitud inferior o una precisión menor. Generalmente, los parámetros de la CF no deben tener atributos de longitud o de precisión superiores a los de los parámetros correspondientes de la SF.

- Las especificaciones del tipo de datos RETURNS de la CF se comparan con las de la SF. El tipo de datos final RETURNS de la SF, tras la conversión que sea, debe ser el mismo o bien ser convertible al tipo de datos RETURNS de la CF. De lo contrario, se produce un error (SQLSTATE 42866).

Tenga en cuenta que esta regla no proporciona ninguna garantía frente a los errores producidos al utilizar la CF. Un valor de resultado que coincide con el tipo de datos y los atributos de longitud y de precisión del tipo de datos RETURNS de la SF, tal vez no sea asignable si el tipo de datos RETURNS de la CF tiene una longitud inferior o una precisión menor. Debe tenerse precaución al especificar el tipo de datos RETURNS de la CF con atributos de precisión o longitud inferiores a los atributos del tipo de datos RETURNS de la SF.

Notas

- La determinación de si un tipo de datos es convertible a otro no tiene en cuenta la longitud, precisión ni escala de los tipos de datos parametrizados, como son CHAR y DECIMAL. Por esta razón, pueden producirse errores al utilizar una función como resultado del intento de conversión de un valor del tipo de datos fuente a un valor del tipo de datos de destino. Por ejemplo, VARCHAR es convertible a DATE, pero si el tipo fuente está realmente definido como VARCHAR(5), al utilizar la función se producirá un error.

CREATE FUNCTION (fuente o plantilla)

- Al elegir los tipos de datos para los parámetros de una función definida por el usuario, tenga en cuenta las reglas para la promoción que afectarán a los valores de entrada (consulte el apartado “Promoción de los tipos de datos” en la página 100). Por ejemplo, una constante que puede utilizarse como un valor de entrada puede tener un tipo de datos incorporado diferente al que se espera y, lo que es más significativo, es posible que no se promueva al tipo de datos que se espera. De acuerdo con las reglas para la promoción, suele aconsejarse la utilización de los siguientes tipos de datos para parámetros:
 - INTEGER en lugar de SMALLINT
 - DOUBLE en lugar de REAL
 - VARCHAR en lugar de CHAR
 - VARGRAPHIC en lugar de GRAPHIC
- La creación de una función con un nombre de esquema que aún no existe dará como resultado la creación implícita de dicho esquema siempre que el ID de autorización de la sentencia tenga la autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN en el esquema se otorga a PUBLIC.
- Para que un servidor federado reconozca una función fuente de datos, la función debe estar correlacionada con una función complementaria en la base de datos federada. Si la base de datos no contiene ninguna función complementaria, el usuario debe crearla y luego definir la correlación.

La función complementaria puede ser una función (escalar o fuente) o una plantilla de función. Si el usuario crea una función y la correlación necesaria, entonces cada vez que se procese una consulta que especifique la función, DB2 (1) compara las estrategias para invocarla con estrategias para invocar la función fuente de datos e (2) invoca la función que se prevé que exigirá menos recursos adicionales.

Si el usuario crea una plantilla de función y la correlación, cada vez que se procesa una consulta donde se especifica la plantilla, DB2 invoca la función de fuente de datos con la que la plantilla está correlacionado, siempre que exista un plan de acceso para invocar esta función. Consulte el manual *Application Development Guide* para obtener más información sobre cómo controlar la sobrecarga de invocar funciones en un sistema federado.

Ejemplos

Ejemplo 1: Algún tiempo después de la creación de la función escalar externa CENTRE de Pellow, otro usuario desea crear una función basada en ella, excepto que esta función está pensada para aceptar solamente argumentos enteros.

```
CREATE FUNCTION MYCENTRE (INTEGER, INTEGER)  
RETURNS FLOAT  
SOURCE PELLOW.CENTRE (INTEGER, FLOAT)
```

CREATE FUNCTION (fuente o plantilla)

Ejemplo 2: Se ha creado un tipo diferenciado, HATSIZE, basándose en el tipo de datos INTEGER incorporado. Sería útil tener una función AVG para calcular el tamaño medio (hatsize) de los distintos departamentos. Esto se lleva a cabo fácilmente de la manera siguiente:

```
CREATE FUNCTION AVG (HATSIZE) RETURNS HATSIZE  
SOURCE SYSIBM.AVG (INTEGER)
```

La creación del tipo diferenciado ha generado la función de conversión necesaria, permitiendo la conversión de HATSIZE en INTEGER para el argumento y de INTEGER a HATSIZE para el resultado de la función.

Ejemplo 3: En un sistema federado, un usuario desea invocar una UDF Oracle que devuelva estadísticas de tabla en forma de valores con comas flotantes de precisión doble. El servidor federado sólo puede reconocer esta función si existe una correlación entre la función y una función complementaria situada en una base de datos federada. Pero esta función complementaria no existe. El usuario decide suministrar una función complementaria en forma de plantilla de función y asignar esta plantilla a un esquema llamado NOVA. El usuario utiliza el siguiente código para registrar la plantilla en el servidor federado; para conocer el código de usuario utilizado para la correlación, vea "Ejemplos" en la página 772.

```
CREATE FUNCTION NOVA.STATS (DOUBLE, DOUBLE)  
RETURNS DOUBLE  
AS TEMPLATE
```

Ejemplo 4: En un sistema federado, un usuario desea invocar una UDF de Oracle que devuelve los importes en dólares que los empleados de una organización determinada ganan en forma de bonos. El servidor federado sólo puede reconocer esta función si existe una correlación entre la función y una función complementaria situada en una base de datos federada. No existe esta función complementaria; por lo tanto, el usuario crea una en forma de plantilla de función. El usuario utiliza el siguiente código para registrar esta plantilla en el servidor federado; para conocer el código de usuario utilizado para la correlación, vea "Ejemplos" en la página 772.

```
CREATE FUNCTION BONUS ()  
RETURNS DECIMAL (8,2)  
AS TEMPLATE
```

CREATE FUNCTION (SQL, escalar, de tabla o de fila)

Esta sentencia se utiliza para definir una función SQL escalar, de tabla o de fila, definida por el usuario. Una *función escalar* devuelve un solo valor cada vez que se invoca y en general es válida donde una expresión SQL sea válida. Una *función de tabla* puede utilizarse en la cláusula FROM y devuelve una tabla. Una *función de fila* puede utilizarse como función de transformación y devuelve una fila.

Invocación

Esta sentencia se puede intercalar en un programa de aplicación o se puede emitir utilizando sentencias de SQL dinámicas. Es una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

Autorización

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

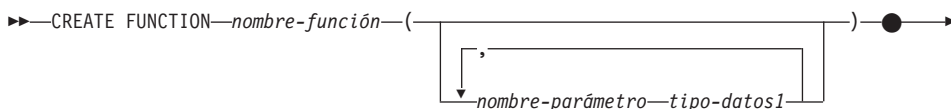
- Autorización SYSADM o DBADM
- Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema de la función no hace referencia a un esquema existente
- Privilegio CREATEIN para el esquema, si el nombre de esquema de la función no hace referencia a un esquema existente.

Si el ID de autorización de la sentencia no tiene autorización SYSADM ni DBADM, y la función identifica una tabla o vista, los privilegios que tiene el ID de autorización de la sentencia (además de los privilegios de grupo) deben incluir SELECT WITH GRANT OPTION para cada tabla o vista identificada.

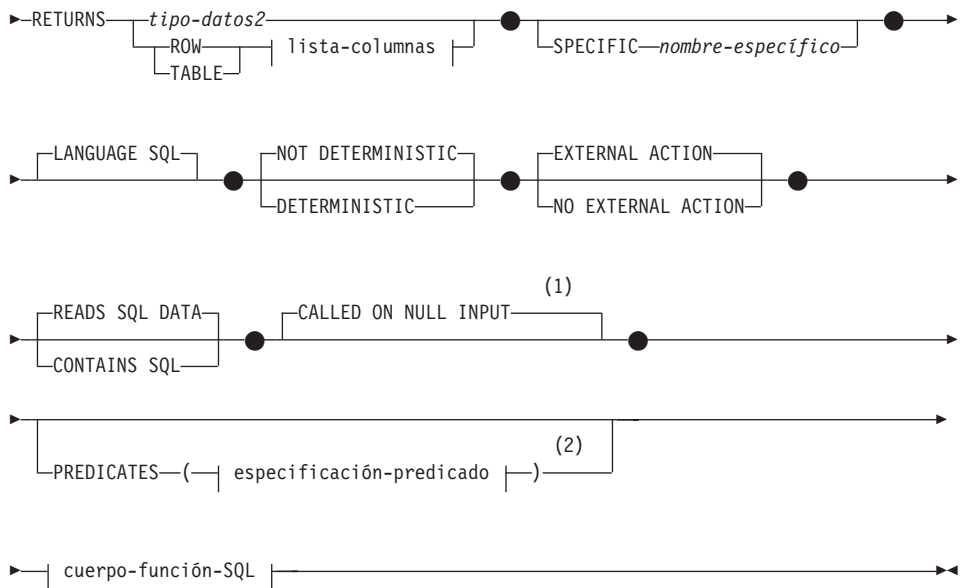
Si el definidor de una función sólo puede crear la función porque tiene autorización SYSADM, se le otorga la autorización implícita DBADM para crear la función.

Si el ID de autorización no tiene la autorización suficiente para realizar la operación, se produce un error (SQLSTATE 42502).

Sintaxis



CREATE FUNCTION (SQL, escalar, de tabla o de fila)



lista-columnas:



cuerpo-función-SQL:



Notas:

- 1 Puede especificarse NULL CALL en lugar de CALLED ON NULL INPUT.
- 2 Sólo es válido si RETURNS especifica un resultado escalar (tipo-datos2)

Descripción

nombre-función

Indica el nombre de la función que se está definiendo. Consiste en el nombre, calificado o no calificado, que designa una función. La forma no calificada del *nombre-función* es un identificador SQL (cuya longitud máxima es de 18). En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto

CREATE FUNCTION (SQL, escalar, de tabla o de fila)

no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objetos no calificados. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL.

El nombre, incluidos los calificadores implícitos y explícitos, junto con el número de parámetros y el tipo de datos de cada parámetro (sin tener en cuenta ningún atributo de longitud, precisión o escala del tipo de datos) no debe identificar una función descrita en el catálogo (SQLSTATE 42723). El nombre no calificado, junto con el número y el tipo de datos de los parámetros, que por supuesto es exclusivo en su esquema, no es necesario que lo sea en todos los esquemas.

Si se especifica un nombre compuesto de dos partes, el *nombre-esquema* no puede empezar por "SYS" (SQLSTATE 42939).

Algunos nombres utilizados como palabras clave en predicados están reservados para su uso por el sistema, y no pueden utilizarse como *nombre-función* (SQLSTATE 42939). Estos nombres son SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH y los operadores de comparación tal como se describe en "Predicado básico" en la página 213.

Se puede utilizar el mismo nombre para más de una función si hay alguna diferencia en la signatura de las funciones. Aunque no hay ninguna prohibición al respecto, una función de tabla externa definida por el usuario no debe tener el mismo nombre que una función incorporada.

nombre-parámetro

Especifica un nombre que es diferente de los nombres de todos los demás parámetros de la función.

tipo-datos1

Especifica el tipo de datos del parámetro:

- Pueden proporcionarse especificaciones y abreviaturas de tipo de datos SQL que puedan especificarse en la definición de *tipo-datos1* de la sentencia CREATE TABLE.
- Se puede especificar REF, pero no tiene un ámbito definido. El sistema no intenta deducir el ámbito del parámetro o resultado. Dentro del cuerpo de la función, se puede utilizar un tipo de referencia en una operación de desreferencia sólo si primero se convierte para que tenga un ámbito. Similarmente, una referencia devuelta por una función SQL se puede utilizar en una operación de desreferencia sólo si primero se convierte para que tenga un ámbito.
- Los tipos de datos LONG VARCHAR y LONG VARGRAPHIC no se pueden utilizar (SQLSTATE 42815).

RETURNS

Esta cláusula obligatoria identifica el tipo de salida de la función.

CREATE FUNCTION (SQL, escalar, de tabla o de fila)

tipo-datos2

Especifica el tipo de datos de la salida.

En esta sentencia, son válidas exactamente las mismas consideraciones que se describieron anteriormente en *tipo-datos1* para parámetros de funciones SQL.

ROW *lista-columns*

Especifica que la salida de la función es una fila individual. Si la función devuelve más de una fila, se produce un error (SQLSTATE 21505). La *lista-columns* debe incluir dos columnas como mínimo (SQLSTATE 428F0).

Una función de fila sólo se puede utilizar como función de transformación para un tipo estructurado (utilizando un tipo estructurado como parámetro y devolviendo sólo tipos base).

TABLE *lista-columns*

Especifica que el resultado de la función es una tabla.

lista-columns

Es la lista de nombres de columnas y tipos de datos devueltos para una función de fila o de tabla.

nombre-columna

Especifica el nombre de esta columna. El nombre no puede estar calificado y no se puede utilizar el mismo nombre para más de una columna de la fila.

tipo-datos3

Especifica el tipo de datos de la columna y puede ser cualquier tipo de datos soportado por un parámetro de la función SQL.

SPECIFIC *nombre-especifico*

Proporciona un nombre exclusivo para la instancia de la función que se está definiendo. Este nombre específico puede utilizarse cuando se utiliza esta función como fuente, al eliminar la función o bien al comentarla. Nunca puede emplearse para invocar a la función. La forma no calificada de *nombre-especifico* es un identificador SQL (cuya longitud máxima es 18). La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL. El nombre, incluido el calificador implícito o explícito, no debe identificar otra instancia de función que exista en el servidor de aplicaciones; de lo contrario se produce un error (SQLSTATE 42710).

El *nombre-especifico* puede ser el mismo que un *nombre-función* ya existente.

Si no se especifica ningún calificador, se emplea el que se haya utilizado para el *nombre-función*. Si se especifica un calificador, debe ser el mismo que el calificador explícito o implícito de *nombre-función*; de lo contrario se produce un error (SQLSTATE 42882).

CREATE FUNCTION (SQL, escalar, de tabla o de fila)

Si no se especifica el *nombre-especifico*, el gestor de bases de datos genera un nombre exclusivo. El nombre exclusivo es SQL seguido de una indicación de fecha y hora en forma de serie caracteres: SQLaammddhhmssxxx.

LANGUAGE SQL

Especifica que la función está escrita en SQL.

DETERMINISTIC o NOT DETERMINISTIC

Esta cláusula opcional especifica si la función siempre devuelve los mismos resultados para los valores de argumento determinados (DETERMINISTIC) o si la función depende de algunos valores de estado que afectan a los resultados (NOT DETERMINISTIC). Es decir, una función DETERMINISTIC siempre debe devolver la misma tabla ante invocaciones sucesivas con entradas idénticas. Se evitan las optimizaciones que aprovechan el hecho de que entradas idénticas siempre produzcan los mismos resultados especificando NOT DETERMINISTIC.

Se debe especificar NOT DETERMINISTIC, de forma explícita o implícita, si el cuerpo de la función accede a un registro especial, o invoca otra función no determinista (SQLSTATE 428C2).

NO EXTERNAL ACTION o EXTERNAL ACTION

Esta cláusula opcional especifica si la función emprende o no alguna acción que cambie el estado de un objeto no gestionado por el gestor de bases de datos. Si se especifica NO EXTERNAL ACTION, el sistema puede utilizar determinadas optimizaciones basadas en que la función no produce ningún efecto externo.

Se debe especificar EXTERNAL ACTION, de forma explícita o implícita, si el cuerpo de la función invoca otra función que tiene una acción externa (SQLSTATE 428C2).

READS SQL DATA o CONTAINS SQL

Indica qué tipo de sentencias de SQL se pueden ejecutar. Debido a que la sentencia de SQL soportada es la sentencia RETURN, la distinción está relacionada con el hecho de si la expresión es una subconsulta o no.

READS SQL DATA

Indica que la función puede ejecutar sentencias de SQL que no modifiquen datos SQL (SQLSTATE 42985). No se pueden utilizar apodos ni funciones de tabla OLEDB en la sentencia de SQL (SQLSTATE 42997).

CONTAINS SQL

Indica que la función puede ejecutar sentencias de SQL que no lean ni modifiquen datos SQL (SQLSTATE 42985).

CALLED ON NULL INPUT

Esta cláusula indica que se invoca la función con independencia de si

CREATE FUNCTION (SQL, escalar, de tabla o de fila)

cualquiera de sus argumentos es nulo. Puede devolver un valor nulo o un valor no nulo. En este caso, la función definida por el usuario debe comprobar si los valores de los argumentos son nulos.

Puede especificarse NULL CALL en lugar de CALLED ON NULL INPUT.

PREDICATES

Para los predicados que hacen uso de esta función, esta cláusula indica quienes pueden explotar las extensiones de índice, y pueden utilizar la cláusula opcional SELECTIVITY para la condición de búsqueda del predicado. Si se especifica la cláusula PREDICATES, la función debe definirse como DETERMINISTIC con NO EXTERNAL ACTION (SQLSTATE 42613).

especificación-predicado

Vea “CREATE FUNCTION (Escalar externa)” en la página 696 para obtener detalles sobre las especificaciones de predicados.

cuerpo-función-SQL

Especifica el cuerpo de la función. En el cuerpo-función-SQL se puede hacer referencia a nombres de parámetros. Los nombres de parámetros pueden calificarse con el nombre de función para evitar referencias ambiguas.

Si el cuerpo-función-SQL es una sentencia dinámica compuesta, deberá contener como mínimo una sentencia RETURN y se deberá ejecutar una sentencia RETURN cuando se llame a la función (SQLSTATE 42632). Si la función es una función de fila o tabla, puede contener sólo una sentencia RETURN que debe ser la última sentencia en el compuesto dinámico (SQLSTATE 429BD).

Para obtener detalles adicionales, consulte el apartado “SQL compuesto (dinámico)” en la página 642 y el apartado “Sentencia RETURN” en la página 1245.

Notas

- La resolución de las llamadas de función dentro del cuerpo de la función se realiza de acuerdo con la vía de función que está vigente para la sentencia CREATE FUNCTION y no cambia una vez creada la función.
- Si una función SQL contiene varias referencias a cualquiera de los registros especiales de fecha u hora, todas las referencias devuelven el mismo valor, y será el mismo valor devuelto por la invocación de registro en la sentencia donde se invocó la función.
- El cuerpo de una función SQL no puede contener una llamada recursiva a sí misma o a otra función o método que la llame, dado que una función de este tipo no puede existir para llamarla.
- Todas las sentencias que crean funciones o métodos aplican las reglas siguientes:

CREATE FUNCTION (SQL, escalar, de tabla o de fila)

- Una función no puede tener la misma signatura que un método (cuando se compara el primer *tipo-parámetro* de la función con el *tipo-sujeto* del método).
- Una función y un método no pueden prevalecer el uno sobre el otro. Esto significa que si la función fuera un método con su primer parámetro como sujeto, no debe invalidar a otro método ni ser invalidado por otro método.
- Debido a que una función no puede prevalecer sobre otra, dos funciones pueden coexistir en situaciones en las que, si fueran métodos, una prevalecería sobre la otra.

Por lo que respecta a la comparación de tipos de parámetros en las reglas anteriores:

- Los nombres de parámetros, longitudes, AS LOCATOR y FOR BIT DATA no se tienen en cuenta.
- Un subtipo se considera que es diferente que su supertipo.

Ejemplos

Ejemplo 1: Definición de una función escalar que devuelve la tangente de un valor utilizando las funciones de seno y coseno existentes.

```
CREATE FUNCTION TAN (X DOUBLE)
  RETURNS DOUBLE
  LANGUAGE SQL
  CONTAINS SQL
  NO EXTERNAL ACTION
  DETERMINISTIC
  RETURN SIN(X)/COS(X)
```

Ejemplo 2: Definición de una función de transformación para el tipo estructurado PERSON.

```
CREATE FUNCTION FROMPERSON (P PERSON)
  RETURNS ROW (NAME VARCHAR(10), FIRSTNAME VARCHAR(10))
  LANGUAGE SQL
  CONTAINS SQL
  NO EXTERNAL ACTION
  DETERMINISTIC
  RETURN VALUES (P..NAME, P..FIRSTNAME)
```

Ejemplo 3: Definición de una tabla de función que muestra los empleados que trabajan en un número de departamento especificado.

```
CREATE FUNCTION DEPTEMPLOYEES (DEPTNO CHAR(3))
  RETURNS TABLE (EMPNO CHAR(6),
                 LASTNAME VARCHAR(15),
                 FIRSTNAME VARCHAR(12))
  LANGUAGE SQL
  READS SQL DATA
  NO EXTERNAL ACTION
  DETERMINISTIC
```

CREATE FUNCTION (SQL, escalar, de tabla o de fila)

```
RETURN
  SELECT EMPNO, LASTNAME, FIRSTNME
         FROM EMPLOYEE
  WHERE EMPLOYEE.WORKDEPT = DEPTEMPLOYEES.DEPTNO
```

Observe que el definidor de esta función debe tener el privilegio `SELECT WITH GRANT OPTION` para la tabla `EMPLOYEE` y que todos los usuarios pueden invocar la función de tabla `DEPTEMPLOYEES`, lo cual les proporciona acceso a los datos de las columnas resultantes correspondientes a cada número de departamento.

CREATE FUNCTION MAPPING

La sentencia CREATE FUNCTION MAPPING se utiliza para:

- Crear una correlación entre una función o una plantilla de función, situadas en una base de datos federada, y una función fuente de datos. La correlación puede asociar la función o plantilla de la de base de datos federada con una función de (1) una fuente de datos especificada o de (2) un rango de fuentes de datos; por ejemplo, todas las fuentes de datos de un tipo y versión en particular.
- Inhabilitar una correlación por omisión entre una función de base de datos federada y una función de fuente de datos.

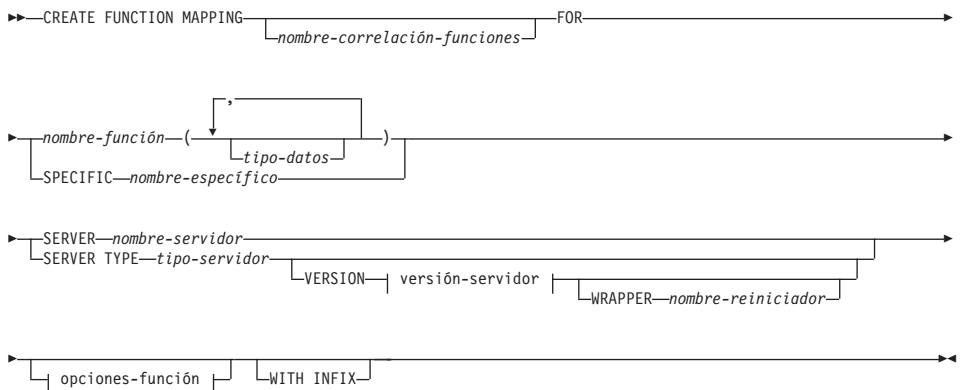
Invocación

Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

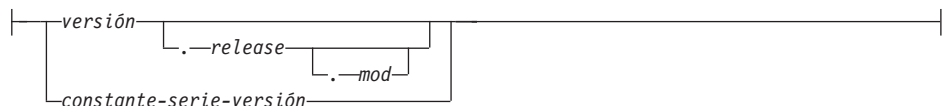
Autorización

El ID de autorización de la sentencia debe tener la autorización SYSADM o DBADM.

Sintaxis

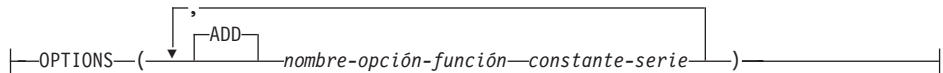


versión-servidor:



CREATE FUNCTION MAPPING

opciones-función:



Descripción

nombre-correlación-funciones

Nombra la correlación de funciones. El nombre no debe identificar ninguna correlación de funciones que ya esté descrita en el catálogo (SQLSTATE 42710).

Si se omite *nombre-correlación-funciones*, se asigna un nombre exclusivo generado por el sistema.

nombre-función

Es el nombre, calificado o no calificado, de la función o plantilla de función desde la que se debe establecer una correlación.

tipo-datos

Para una función o plantilla de función que tenga algún parámetro de entrada, *tipo-datos* especifica el tipo de datos de dicho parámetro. El *tipo de datos* no puede ser `LONG VARCHAR`, `LONG VARGRAPHIC`, `DATALINK`, un tipo de objeto grande (LOB) ni un tipo definido por el usuario.

SPECIFIC *nombre-especifico*

Identifica la función o plantilla de función desde la que se debe establecer una correlación. Especifique *nombre-especifico* si la función o plantilla de función no tiene un *nombre-función* exclusivo en la base de datos federada.

SERVER *nombre-servidor*

Designa la fuente de datos donde reside la función con la que se establece la correlación.

TYPE *tipo-servidor*

Identifica el tipo de fuente de datos que contiene la función con la que se está correlacionando.

VERSION

Identifica la versión de la fuente de datos indicada por *tipo-servidor*.

versión

Especifica el número de versión. *versión* debe ser un entero.

release

Especifica el número de release de la versión indicada por *versión*. *release* debe ser un entero.

mod

Especifica el número de la modificación del release indicado por *release*. *mod* debe ser un entero.

constante-serie-versión

Especifica la designación completa de la versión. La *constante-serie-versión* puede ser un solo valor (por ejemplo, '8i'); o puede ser los valores concatenados de *versión*, *release* y, si se aplica, *mod* (por ejemplo, '8.0.3').

WRAPPER *nombre-reiniciador*

Especifica el nombre del reiniciador que el servidor federado utiliza para interactuar con las fuentes de datos del tipo y versión indicados por *tipo-servidor* y *versión-servidor*.

OPTIONS

Indica las opciones de correlación de funciones que se han de habilitar. Consulte "Opciones de correlación de funciones" en la página 1414 para ver las descripciones de *nombre-opción-función* y sus valores.

ADD

Habilita una o varias opciones de correlación de funciones.

nombre-opción-función

Nombra una opción de correlación de funciones que se aplica a la correlación de funciones o a la función de fuente de datos incluida en la correlación.

constante-serie

Especifica el valor para *nombre-opción-función* como una constante de serie de caracteres.

WITH INFIX

Especifica que la función de fuente de datos se ha de generar en formato infijo.

Notas

- Una función o plantilla de función de base de datos federada puede correlacionarse con una función de fuente de datos si:
 - La función o plantilla de la base de datos federada tiene el mismo número de parámetros de entrada que la función de fuente de datos.
 - Los tipos de datos que están definidos para la función o plantilla federados son compatibles con los tipos de datos correspondientes que están definidos para la función de fuente de datos.
- Si una petición distribuida hace referencia a una función DB2 que se correlaciona con una función de fuente de datos, el optimizador desarrolla estrategias para invocar cualquiera de las dos funciones cuando se procesa la petición. Se invoca la función DB2 si para ello se necesita menos actividad general que para invocar la función de fuente de datos. De lo

CREATE FUNCTION MAPPING

contrario, si para invocar la función DB2 se necesita más actividad general, se invoca la función de fuente de datos.

- Si una petición distribuida hace referencia a una plantilla de función DB2 que se correlaciona con una función de fuente de datos, sólo se puede invocar la función de fuente de datos cuando se procesa la petición. La plantilla no puede invocarse porque no tiene ningún código ejecutable.
- Las correlaciones de funciones por omisión pueden pasar a inoperativas inhabilitándolas (no se pueden desactivar). Para inhabilitar la correlación de funciones, codifique la sentencia CREATE FUNCTION MAPPING de modo que especifique el nombre de la función DB2 en la correlación y establezca la opción DISABLE en 'Y'.
- Las funciones del esquema SYSIBM no tienen ningún nombre específico. Para alterar temporalmente la correlación de funciones por omisión para una función del esquema SYSIBM, especifique *nombre-función* con el calificador SYSIBM y un nombre de función (por ejemplo, LENGTH).
- Una sentencia CREATE FUNCTION MAPPING de una unidad de trabajo (UOW) determinada no se puede procesar bajo ninguna de las condiciones siguientes:
 - La sentencia hace referencia a una sola fuente de datos y la UOW ya incluye una sentencia SELECT que hace referencia a un apodo para una tabla o una vista de esta fuente de datos.
 - La sentencia hace referencia a una categoría de fuentes de datos (por ejemplo, todas las fuentes de datos de un tipo y versión específicos) y la UOW ya incluye una sentencia SELECT que hace referencia a un apodo para una tabla o una vista de una de estas fuentes de datos.

Ejemplos

Ejemplo 1: Correlacione una plantilla de función con una UDF a la que pueden acceder todas las fuentes de datos Oracle. La plantilla se llama STATS y pertenece a un esquema llamado NOVA. La UDF de Oracle se llama STATISTICS y pertenece a un esquema llamado STAR.

```
CREATE FUNCTION MAPPING MY_ORACLE_FUN1
FOR NOVA.STATS ( DOUBLE, DOUBLE )
SERVER TYPE ORACLE
OPTIONS ( REMOTE_NAME 'STAR.STATISTICS' )
```

Ejemplo 2: Correlacione una plantilla de función llamado BONUS con una UDF, que también se llama BONUS y que se utiliza en una fuente de datos Oracle llamada ORACLE1.

```
CREATE FUNCTION MAPPING MY_ORACLE_FUN2
FOR BONUS()
SERVER ORACLE1
OPTIONS ( REMOTE_NAME 'BONUS' )
```

Ejemplo 3: Suponga que existe una correlación de funciones por omisión entre la función del sistema WEEK que está definida en la base de datos federada y una función similar que está definida en las fuentes de datos Oracle. Cuando se procesa una consulta que pide datos Oracle y que hace referencia a WEEK, se invocará WEEK o su equivalente de Oracle, dependiendo de cuál estime el optimizador que necesita menos actividad general. El DBA desea averiguar cómo afectaría al rendimiento si sólo se invocase WEEK para dichas consultas. Para asegurarse de que se invoca WEEK cada vez, el DBA debe inhabilitar la correlación.

```
CREATE FUNCTION MAPPING  
FOR SYSFUN.WEEK(INT)  
TYPE ORACLE  
OPTIONS ( DISABLE 'Y' )
```

Ejemplo 4: Correlacione la función UCASE(CHAR) con una UDF que se utiliza en una fuente de datos Oracle llamada ORACLE2. Incluya el número estimado de instrucciones por invocación de la UDF Oracle.

```
CREATE FUNCTION MAPPING MY_ORACLE_FUN4  
FOR SYSFUN.UCASE(CHAR)  
SERVER ORACLE2  
OPTIONS  
  ( REMOTE_NAME 'UPPERCASE',  
    INSTS_PER_INVOC '1000' )
```

CREATE INDEX

CREATE INDEX

La sentencia CREATE INDEX se utiliza para crear:

- Un índice en una tabla de DB2
- Una especificación de índice: metadatos que indican al optimizador que una tabla fuente tiene un índice

Invocación

Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

Autorización

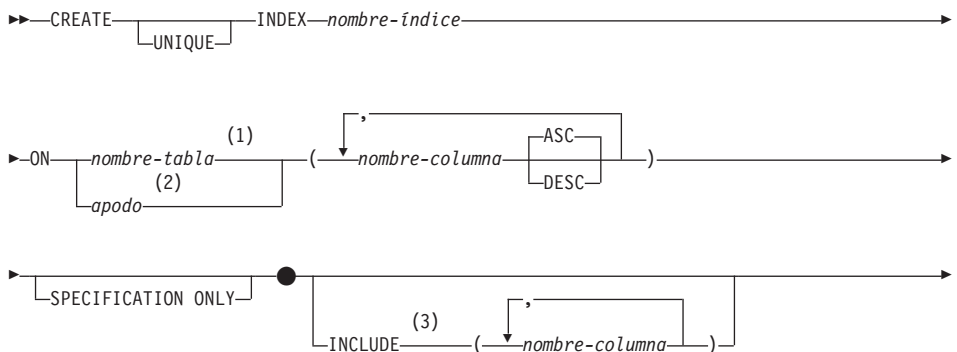
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

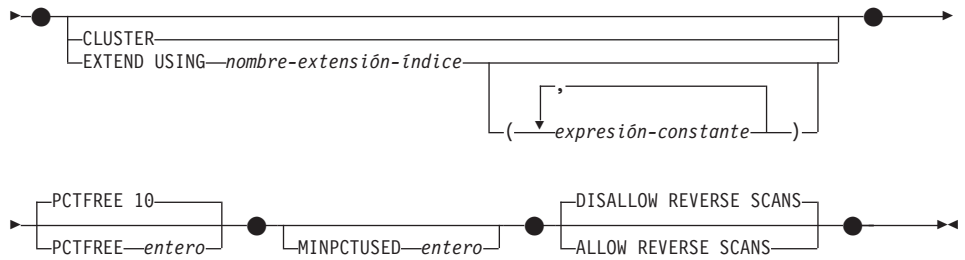
- Autorización SYSADM o DBADM.
- Uno de los siguientes:
 - Privilegio CONTROL para la tabla
 - Privilegio INDEX para la tabla

y uno de estos:

- Autorización IMPLICIT_SCHEMA para la base de datos, si no existe el nombre de esquema implícito o explícito del índice.
- Privilegio CREATEIN para el esquema, si el nombre de esquema del índice hace referencia a un esquema existente.

Sintaxis





Notas:

- 1 En un sistema federado, el *nombre-tabla* debe identificar una tabla en la base de datos federada. No puede identificar una tabla de fuente de datos.
- 2 Si se especifica *apodo*, la sentencia CREATE INDEX creará una especificación de índice. INCLUDE, CLUSTER, PCTFREE, MINPCTUSED, DISALLOW REVERSE SCANS y ALLOW REVERSE SCANS no se pueden especificar.
- 3 La cláusula INCLUDE sólo puede especificarse si se especifica UNIQUE.

Descripción

UNIQUE

Si se especifica ON *nombre-tabla*, UNIQUE evita que la tabla contenga dos o más filas con el mismo valor de la clave de índice. La unicidad de valores se aplica al final de la sentencia de SQL que actualiza filas o inserta nuevas filas. Para obtener detalles consulte el “Apéndice J. Interacción de desencadenantes y restricciones” en la página 1455.

La exclusividad también se comprueba durante la ejecución de la sentencia CREATE INDEX. Si la tabla ya contiene filas con valores de clave duplicados, no se crea el índice.

Cuando se utiliza UNIQUE, los valores nulos se tratan como cualquier otro valor. Por ejemplo, si la clave es una sola columna que puede contener valores nulos, esa columna no puede contener más de un valor nulo.

Si se especifica la opción UNIQUE y la tabla tiene una clave de particionamiento, las columnas de la clave de índice deben ser un superconjunto de la clave de particionamiento. Es decir, las columnas especificadas para una clave de índice de unicidad deben incluir todas las columnas de la clave de particionamiento (SQLSTATE 42997).

Si se especifica ON *apodo*, UNIQUE se debe especificar sólo si los datos para la clave de índice contienen valores exclusivos para cada fila de la tabla fuente de datos. No se comprobará la exclusividad.

CREATE INDEX

El nombre-tabla no puede ser una tabla temporal declarada (SQLSTATE 42995).

INDEX *nombre-índice*

Nombra el índice o la especificación de índice. El nombre, incluido el calificador implícito o explícito, no debe designar un índice o especificación de índice que esté descrito en el catálogo. El calificador no debe ser SYSIBM, SYSCAT, SYSFUN ni SYSSTAT (SQLSTATE 42939)

ON *nombre-tabla* **o** *apodo*

El *nombre-tabla* designa una tabla para la que se creará un índice. La tabla debe ser una tabla base (no una vista) o una tabla de resumen descrita en el catálogo. El nombre de tabla no debe ser una tabla de catálogo (SQLSTATE 42832) ni una tabla temporal declarada (SQLSTATE 42995). Si se especifica UNIQUE y *nombre-tabla* es una tabla con tipo, no debe ser una subtabla (SQLSTATE 429B3). Si se especifica UNIQUE, *nombre-tabla* no puede ser una tabla de resumen (SQLSTATE 42809).

apodo es el apodo en el que una especificación de índice se creará. El *apodo* hace referencia a la tabla de fuente de datos cuyo índice se describe mediante la especificación de índice o la vista de fuente de datos que se basa en esa tabla. El *apodo* debe estar listado en el catálogo.

nombre-columna

Para un índice, *nombre-columna* identifica una columna que ha de formar parte de la clave de índice. Para una especificación de índice, *nombre-columna* es el nombre que el servidor federado utiliza para hacer referencia a una columna de una tabla fuente de datos.

Cada *nombre-columna* debe ser un nombre no calificado que identifique a una columna de la tabla. Pueden especificarse 16 columnas como máximo. Si *nombre-tabla* es una tabla con tipo, pueden especificarse 15 columnas como máximo. Si *nombre-tabla* es una subtabla, debe entrarse como mínimo un *nombre-columna* en la subtabla, es decir, no heredada de una supertabla (SQLSTATE 428DS). No puede haber repetido ningún *nombre-columna* (SQLSTATE 42711).

La suma de los atributos de longitud de las columnas especificadas no debe ser mayor que 1024. Si *nombre-tabla* es una tabla con tipo, la longitud de la clave de índice debe ser todavía 4 bytes menos.

Observe que esta longitud puede ser menor debido a la actividad general del sistema, que varía según el tipo de datos de la columna y según si puede contener nulos. Vea "Cuentas de bytes" en la página 885 para obtener más información acerca de la actividad general que afecta a este límite.

La longitud cualquier columna individual no debe ser mayor que 255 bytes. No puede utilizarse como parte de un índice ninguna columna LOB, columna DATALINK ni una columna de un tipo diferenciado

basado en un LOB o DATALINK, aunque el atributo de longitud de la columna sea lo bastante pequeño como para caber dentro del límite de 255 bytes (SQLSTATE 42962). Las columnas de tipo estructurado sólo se pueden especificar si también se especifica la cláusula EXTEND USING (SQLSTATE 42962). Si se especifica la cláusula EXTEND USING, sólo se puede especificar una sola columna, que debe ser de tipo estructurado o de un tipo diferenciado no basado en un LOB, DATALINK, LONG VARCHAR ni LONG VARCHAR (SQLSTATE 42997).

ASC

Especifica que las entradas de índice se deben mantener en el orden ascendente de los valores de columna; este es el valor por omisión. ASC no se puede especificar para índices que están definidos con EXTEND USING (SQLSTATE 42601).

DESC

Especifica que las entradas de índice se deben mantener en el orden descendente de los valores de columna. DESC no se puede especificar para índices que están definidos con EXTEND USING (SQLSTATE 42601).

SPECIFICATION ONLY

Indica que esta sentencia será utilizada para crear una especificación de índice que se suscribe a la tabla de fuente de datos a la que se hace referencia mediante *apodo*. Se debe especificar SPECIFICATION ONLY si se especifica *apodo* (SQLSTATE 42601). No se puede especificar si se especifica *nombre-tabla* (SQLSTATE 42601).

INCLUDE

Esta palabra clave introduce una cláusula que especifica columnas adicionales a añadir al conjunto de columnas de la clave de índice. Las columnas incluidas con esta cláusula no se utilizan para imponer la exclusividad. Estas columnas incluidas pueden mejorar el rendimiento de algunas consultas mediante el acceso de sólo índice. Las columnas deben ser diferentes de las columnas utilizadas para imponer la exclusividad (SQLSTATE 42711). Los límites para el número de columnas y la suma de los atributos de longitud se aplican a todas las columnas del índice y la clave de unicidad.

nombre-columna

Identifica una columna que se incluye en el índice, pero que no forma parte de la clave de índice de unicidad. Se aplican las mismas normas que se han definido para las columnas de la clave de índice de unicidad. Pueden especificarse las palabras clave ASC o DESC después del nombre-columna, pero no tienen efecto sobre el orden.

INCLUDE no se puede especificar para índices que están definidos con EXTEND USING, o si está especificado *apodo* (SQLSTATE 42601).

CREATE INDEX

CLUSTER

Especifica que el índice es el índice de agrupación de la tabla. El factor de agrupación de un índice de agrupación se mantiene o se mejora dinámicamente cuando los datos se insertan en la tabla asociada al intentar insertar filas nuevas físicamente cerca de las filas para las que los valores de clave de este índice están en el mismo rango. Sólo puede existir un índice de agrupación para una tabla, por lo que no puede especificarse CLUSTER si se utiliza en la definición de cualquier índice existente en la tabla (SQLSTATE 55012). No puede crearse un índice de agrupación en una tabla que esté definida para utilizar la modalidad APPEND (SQLSTATE 428D8).

CLUSTER no está permitido si *apodo* está especificado (42601).

EXTEND USING *nombre-extensión-índice*

Designa la *extensión-índice* utilizada para gestionar el índice. Si se especifica esta cláusula, debe haber un solo *nombre-columna* especificado y esa columna debe ser un tipo estructurado o un tipo diferenciado (SQLSTATE 42997). El *nombre-extensión-índice* debe ser una extensión de índice descrita en el catálogo (SQLSTATE 42704). Para un tipo diferenciado, la columna debe coincidir exactamente con el tipo del correspondiente parámetro de clave fuente de la extensión de índice. Para una columna de tipo estructurado, el tipo del correspondiente parámetro de clave fuente debe ser el mismo tipo o un supertipo del tipo de columna (SQLSTATE 428E0).

expresión-constante

Designa valores para los argumentos necesarios de la extensión de índice. Cada expresión debe ser un valor constante cuyo tipo de datos coincide exactamente con el tipo de datos definido de los correspondientes parámetros de la extensión de índice, incluidas la longitud o precisión, y la escala (SQLSTATE 428E0).

PCTFREE *entero*

Especifica qué porcentaje de cada página de índice se va a dejar como espacio libre cuando se cree el índice. La primera entrada de una página se añade sin restricciones. Cuando se colocan entradas adicionales en una página de índice, en cada página se deja, como mínimo, un *entero* como porcentaje de espacio libre. El valor de *entero* entra en un rango que va de 0 a 99. No obstante, si se especifica un valor superior a 10, sólo se dejará un 10 por ciento de espacio libre en las páginas sin hoja. El valor por omisión es 10.

PCTFREE no está permitido si *apodo* está especificado (SQLSTATE 42601).

MINPCTUSED *entero*

Indica si los índices se reorganizan en línea y el porcentaje mínimo de espacio utilizado en una página terminal de índice. Si, después de suprimir una clave de una página terminal de índice, el porcentaje de

espacio utilizado en la página es un porcentaje igual o mayor que *entero*, se intentará fusionar las claves restantes de la página con las de una página vecina. Si hay espacio suficiente en una de estas páginas, se realiza la fusión y se elimina una de las páginas. El valor de *entero* puede ir de 0 a 99. Sin embargo, se recomienda un valor de 50 o inferior por motivos de rendimiento.

MINPCTUSED no está permitido si *apodo* está especificado (42601).

DISALLOW REVERSE SCANS

Especifica que un índice sólo permite búsquedas hacia adelante o en el orden definido al crear el índice. Éste es el valor por omisión.

DISALLOW REVERSE SCANS no está permitido si *apodo* está especificado (42601).

ALLOW REVERSE SCANS

Especifica que un índice permite búsquedas hacia delante y hacia atrás; es decir, en el orden definido al crear el índice y en el orden opuesto.

ALLOW REVERSE SCANS no está permitido si *apodo* está especificado (42601).

Normas

- La sentencia CREATE INDEX fallará (SQLSTATE 01550) si se intenta crear un índice que coincida con un índice ya existente. Dos descripciones de índice se consideran duplicadas si:
 - el conjunto de columnas (las columnas de la clave y de inclusión) con su orden en el índices el mismo que el de un índice existente Y
 - los atributos de orden son los mismos Y
 - tanto el índice preexistente como el nuevo no son exclusivos O BIEN el índice que existía anteriormente es exclusivo Y
 - si tanto el índice preexistente como el nuevo son exclusivos, las columnas de clave del índice que se crean son las mismas que las de la clave del índice que existía anteriormente o son un superconjunto de éstas.

Notas

- Si la tabla nombrada ya contiene datos, CREATE INDEX crea las entradas de índice de la misma. Si la tabla todavía no contiene datos, CREATE INDEX crea una descripción del índice; las entradas del índice se crean al insertar los datos en la tabla.
- Una vez se ha creado el índice y se han cargado los datos en la tabla, es aconsejable emitir el mandato RUNSTATS. (Consulte el manual *Consulta de mandatos* para obtener información acerca de RUNSTATS.) El mandato RUNSTATS actualiza las estadísticas que se reúnen en las tablas de bases de datos, las columnas y los índices. Estas estadísticas sirven para determinar

CREATE INDEX

la mejor vía de acceso a las tablas. Mediante la emisión del mandato RUNSTATS, el gestor de bases de datos puede determinar las características del nuevo índice.

- La creación de un índice con un nombre de esquema que no existe todavía dará como resultado la creación implícita del esquema siempre que el ID de autorización de la sentencia tenga la autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN en el esquema se otorga a PUBLIC.
- El optimizador puede recomendar índices antes de crear el índice actual. Consulte el “SET CURRENT EXPLAIN MODE” en la página 1153 para obtener más detalles.
- Si una especificación de índice se está definiendo para una tabla de fuente de datos, el nombre de la especificación de índice no tiene que coincidir con el nombre del índice.
- El optimizador utiliza las especificaciones de índice para mejorar el acceso a las tablas de fuente de datos a las que las especificaciones se suscriben.
- Para obtener más información acerca de las especificaciones de índice, consulte el apartado “Especificaciones de índice” en la página 65.

Ejemplos

Ejemplo 1: Cree un índice denominado UNIQUE_NAM en la tabla PROJECT. La finalidad de este índice consiste en garantizar que en la misma tabla no habrá dos entradas que tengan el mismo valor para el nombre del proyecto (PROJNAME). Las entradas de índice han de estar en orden ascendente.

```
CREATE UNIQUE INDEX UNIQUE_NAM  
ON PROJECT (PROJNAME)
```

Ejemplo 2: Cree un índice denominado JOB_BY_DPT en la tabla EMPLOYEE. Disponga las entradas de índice en orden ascendente por el título de los trabajos (JOB) dentro de cada departamento.

```
CREATE INDEX JOB_BY_DPT  
ON EMPLOYEE (WORKDEPT, JOB)
```

Example 3: El apodo EMPLOYEE hace referencia a la tabla de fuente de datos llamada CURRENT_EMP. Después de que este apodo se creó, se definió un índice en CURRENT_EMP. Las columnas escogidas para la clave de índice fueron WORKDEBT y JOB. Cree una especificación de índice que describa a este índice. Mediante esta especificación, el optimizador sabrá que el índice existe y cuál es su clave. Con esta información, el optimizador puede mejorar su estrategia de acceso a la tabla.

```
CREATE UNIQUE INDEX JOB_BY_DEPT  
ON EMPLOYEE (WORKDEPT, JOB)  
SPECIFICATION ONLY
```

Ejemplo 4: Creación de un tipo de índice ampliado llamado SPATIAL_INDEX en una columna de tipo estructurado. La descripción para la extensión de índice GRID_EXTENSION se utiliza para mantener SPATIAL_INDEX. El literal se proporciona a GRID_EXTENSION para crear el tamaño de la cuadrícula de índice. Para conocer una definición de las extensiones de índice, vea "CREATE INDEX EXTENSION" en la página 782.

```
CREATE INDEX SPATIAL_INDEX ON CUSTOMER (LOCATION)  
EXTEND USING (GRID_EXTENSION (x'000100100010001000400010'))
```

CREATE INDEX EXTENSION

CREATE INDEX EXTENSION

La sentencia `CREATE INDEX EXTENSION` crea un objeto de extensión para utilizar con índices de tablas que tienen columnas de tipo estructurado o de tipo diferenciado.

Invocación

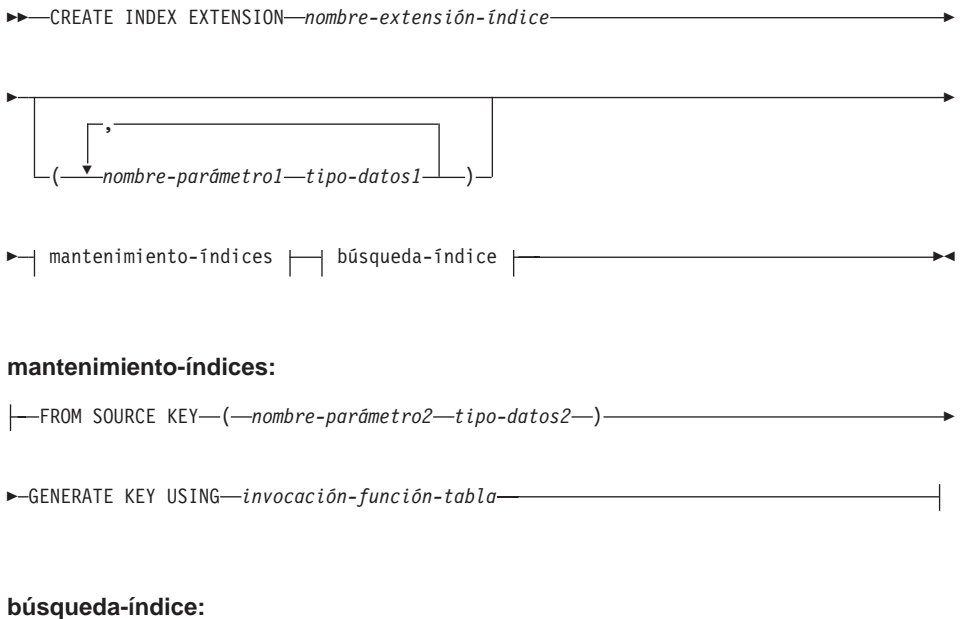
Esta sentencia puede intercalarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace `DYNAMICRULES BIND`, la sentencia no se puede preparar de forma dinámica (`SQLSTATE 42509`).

Autorización

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización `SYSADM` o `DBADM`
- Autorización `IMPLICIT_SCHEMA` para la base de datos (si el nombre de esquema de la extensión de índice no hace referencia a un esquema existente)
- Privilegio `CREATEIN` para el esquema (si el nombre de esquema de la extensión de índice hace referencia a un esquema existente)

Sintaxis



|—WITH TARGET KEY—(—*nombre-parámetro3*—*tipo-datos3*—)

►SEARCH METHODS—| *definición-método-búsqueda* |

definición-método-búsqueda:

|—WHEN—*nombre-método*—(—*nombre-parámetro4*—*tipo-datos4*—)

►RANGE THROUGH—*invocación-función-productora-rango*—

►FILTER USING—| *invocación-función-filtrado-índice* |
 | *expresión-case* |

Descripción

nombre-extensión-índice

Designa la extensión de índice. El nombre, incluido el calificador implícito o explícito, no debe identificar una extensión de índice descrita en el catálogo. Si se especifica un *nombre-extensión-índice* de dos partes, el nombre de esquema no puede empezar por "SYS"; de lo contrario se produce un error (SQLSTATE 42939).

nombre-parámetro1

Identifica un parámetro que se pasa a la extensión de índice, al ejecutar CREATE INDEX, para definir el comportamiento de la extensión de índice. El parámetro que se pasa a la extensión de índice se llama *parámetro de instancia*, pues ese valor define una nueva instancia de una extensión de índice.

nombre-parámetro1 debe ser exclusivo dentro de la definición de la extensión de índice. No se permiten más de 90 parámetros. Si se excede este límite, se produce un error (SQLSTATE 54023).

tipo-datos1

Especifica el tipo de datos de cada parámetro. Debe especificarse una entrada de la lista para cada parámetro que la extensión de índice espera recibir. Los únicos tipos de datos SQL que se pueden especificar son los que se pueden utilizar como constantes, tales como VARCHAR, INTEGER, DECIMAL, DOUBLE o VARGRAPHIC

CREATE INDEX EXTENSION

(SQLSTATE 429B5). Vea “Constantes” en la página 129 para obtener más información sobre constantes. El valor de parámetro que la extensión de índice recibe al ejecutarse CREATE INDEX debe coincidir exactamente con *tipo-datos1*, incluida la longitud, precisión y escala (SQLSTATE 428E0).

mantenimiento-índices

Especifica cómo se realiza el mantenimiento de las claves de índice de una columna de tipo estructurado o de tipo diferenciado. El mantenimiento de índices es el proceso de transformar la columna fuente en una clave destino. El proceso de transformación se define utilizando una función de tabla que previamente se ha definido en la base de datos.

FROM SOURCE KEY (*nombre-parámetro2* *tipo-datos2*)

Especifica un tipo de datos estructurado o tipo diferenciado para la columna de la clave fuente que está soportada por esta extensión de índice.

nombre-parámetro2

Identifica el parámetro que está asociado a la columna de la clave fuente. Una columna de clave fuente es la columna de clave de índice (definida en la sentencia CREATE INDEX) que tiene el mismo tipo de datos que *tipo-datos2*.

tipo-datos2

Especifica el tipo de datos de *nombre-parámetro2*. *tipo-datos2* debe ser un tipo estructurado definido por el usuario o un tipo diferenciado que no esté basado en LOB, DATALINK, LONG VARCHAR ni LONG VARGRAPHIC (SQLSTATE 42997). Cuando la extensión de índice se asocia con el índice al ejecutar CREATE INDEX, el tipo de datos de la columna de clave de índice debe:

- coincidir exactamente con *tipo-datos2* si es un tipo diferenciado; o bien
- ser el mismo tipo o un subtipo de *tipo-datos2* si es un tipo estructurado

De lo contrario se produce un error (SQLSTATE 428E0).

GENERATE KEY USING *invocación-función-tabla*

Especifica cómo se genera la clave de índice utilizando una función de tabla definida por el usuario. Se pueden crear varias entradas de índice para una clave fuente individual. No se puede duplicar una entrada de índice a partir de una clave fuente individual (SQLSTATE 22526). La función puede utilizar *nombre-parámetro1*, *nombre-parámetro2* o una constante como argumentos. Si el tipo de datos de *nombre-parámetro2* es un tipo de datos estructurado, sólo pueden utilizarse en sus argumentos los métodos observadores de ese tipo estructurado (SQLSTATE 428E3). La salida de la función GENERATE

KEY se debe especificar en la especificación TARGET KEY. La salida de la función también se puede utilizar como entrada de la función de filtrado de índice que se especifica en la cláusula FILTER USING.

La función utilizada en *invocación-función-tabla* debe cumplir estas condiciones:

1. Su resolución debe producir una función de tabla (SQLSTATE 428E4)
2. No debe estar definida con LANGUAGE SQL (SQLSTATE 428E4)
3. No debe estar definida como NOT DETERMINISTIC (SQLSTATE 428E4) ni como EXTERNAL ACTION (SQLSTATE 428E4)
4. No debe tener un tipo de datos estructurado, LOB, DATALINK, LONG VARCHAR ni LONG VARGRAPHIC (SQLSTATE 428E3) en el tipo de datos de los parámetros, con la excepción de los métodos observadores generados por el sistema.
5. No debe incluir una subconsulta (SQLSTATE 428E3).
6. Debe devolver columnas cuyos tipos de datos siguen las restricciones para los tipos de datos de columnas de un índice definido sin la cláusula EXTEND USING.

Si un argumento invoca otra operación o rutina, debe ser un método observador (SQLSTATE 428E3).

búsqueda-índice

Especifica cómo se realiza la búsqueda proporcionando una correlación de los argumentos de búsqueda con rangos de búsqueda.

WITH TARGET KEY

Especifica los parámetros de clave destino que son la salida de la función generadora de índices especificada en la cláusula GENERATE KEY USING.

nombre-parámetro3

Identifica el parámetro asociado a una clave destino determinada. *nombre-parámetro3* corresponde a las columnas de la tabla RETURNS tal como está especificado en la función de tabla de la cláusula GENERATE KEY USING. El número de parámetros especificados debe coincidir con el número de columnas devueltas por esa función de tabla (SQLSTATE 428E2).

tipo-datos3

Especifica el tipo de datos de cada *nombre-parámetro3* correspondiente. *tipo-datos3* debe coincidir exactamente con el tipo de datos de las columnas de salida correspondientes de la tabla RETURNS, tal como está especificado en la función de tabla de la cláusula GENERATE KEY USING (SQLSTATE 428E2), incluida la longitud, precisión y tipo.

CREATE INDEX EXTENSION

SEARCH METHODS

Presenta los métodos de búsqueda que están definidos para el índice.

definición-método-búsqueda

Especifica las características del método de la búsqueda por índice. Consta de un nombre de método, los argumentos de búsqueda, una función productora de rangos y una función opcional de filtrado de índice.

WHEN *nombre-método*

Es el nombre de un método de búsqueda. Es un identificador SQL que está asociado con el nombre de método especificado en la regla de explotación de índice (situada en la cláusula PREDICATES de una función definida por el usuario). El *nombre-método-búsqueda* puede estar referenciado por una sola cláusula WHEN en la definición del método de búsqueda (SQLSTATE 42713).

nombre-parámetro4

Identifica el parámetro de un argumento de búsqueda. Estos nombres se utilizan en las cláusulas RANGE THROUGH y FILTER USING.

tipo-datos4

Es el tipo de datos asociado a un parámetro de búsqueda.

RANGE THROUGH *invocación-función-productora-rangos*

Especifica una función de tabla externa que produce rangos de búsqueda. Esta función utiliza *nombre-parámetro1*, *nombre-parámetro4* o una constante como argumentos y devuelve un conjunto de rangos de búsqueda.

La función de tabla utilizada en *invocación-función-productora-rangos* debe cumplir estas condiciones:

1. Su resolución debe producir una función de tabla (SQLSTATE 428E4)
2. No debe incluir una subconsulta (SQLSTATE 428E3) ni una función SQL (SQLSTATE 428E4) en sus argumentos
3. No debe estar definida con LANGUAGE SQL (SQLSTATE 428E4)
4. No debe estar definida como NOT DETERMINISTIC ni como EXTERNAL ACTION (SQLSTATE 428E4)
5. El número y tipos de los resultados de esta función deben estar relacionados con los resultados de la función de tabla especificada en el cláusula GENERATE KEY USING, del modo siguiente (SQLSTATE 428E1):
 - Devuelven un número máximo de columnas igual al doble de las devueltas por la función de transformación de claves
 - Tienen un número par de columnas, donde la primera mitad de las columnas devueltas definen el inicio del rango (valores de

clave de inicio) y la segunda mitad de las columnas devueltas definen el final del rango (valores de clave de parada)

- Cada columna de clave de inicio tiene el mismo tipo que la correspondiente columna de clave de parada
- Cada columna de clave de inicio tiene el mismo tipo que la correspondiente columna de la función de transformación de claves.

Más concretamente, sean $a_1:t_1, \dots, a_n:t_n$ las columnas resultantes y los tipos de datos de la función de transformación. Las columnas resultantes de la *invocación-función-productora-rangos* deben ser $b_1:t_1, \dots, b_m:t_m, c_1:t_1, \dots, c_m:t_m$, donde $m \leq n$ y las columnas "b" son las columnas de clave de inicio y las columnas "c" son las columnas de clave de parada.

Cuando la *invocación-función-productora-rangos* devuelve un valor nulo como valor de clave de inicio o de parada, la semántica no está definida.

FILTER USING

Permite especificar una función externa o expresión CASE para filtrar las entradas de índice resultantes de aplicar la función productora de rangos.

invocación-función-filtrado-índice

Especifica una función externa para el filtrado de entradas de índice. Esta función utiliza el *nombre-parámetro1*, *nombre-parámetro3*, *nombre-parámetro4* o una constante como argumentos (SQLSTATE 42703) y devuelve un entero (SQLSTATE 428E4). Si el valor devuelto es 1, la fila correspondiente a la entrada de índice se recupera de la tabla. En otro caso, la entrada de índice no es objeto de más proceso.

Si no se especifica esta opción, no se realiza el filtrado de índice.

La función utilizada en *invocación-función-filtrado-índice* debe cumplir estas condiciones:

1. No debe estar definida con LANGUAGE SQL (SQLSTATE 429B4)
2. No debe estar definida como NOT DETERMINISTIC ni como EXTERNAL ACTION (SQLSTATE 42845)
3. No debe tener un tipo de datos estructurado para ninguno de los parámetros (SQLSTATE 428E3)
4. No debe incluir una subconsulta (SQLSTATE 428E3)

Si un argumento invoca otra función o método, esas cuatro reglas también se aplican a la función o método anidados. Sin embargo, los métodos observadores generados por el sistema pueden utilizarse como argumentos de la función de filtro (o de cualquier función o

CREATE INDEX EXTENSION

método utilizado como argumento), siempre que el resultado de evaluar el argumento sea un tipo de datos incorporado.

expresión-case

Especifica una expresión CASE para el filtrado de entradas de índice. Se puede utilizar *nombre-parámetro1*, *nombre-parámetro3*, *nombre-parámetro4* o una constante (SQLSTATE 42703) en la *cláusula-when-búsqueda* y en la *cláusula-when-simple*. Como *expresión-resultante* se puede utilizar una función externa con las reglas especificadas en FILTER USING *invocación-función-filtrado-índice*. Cualquier función referenciada en la *expresión-case* debe también seguir las cuatro reglas descritas para *invocación-función-filtrado-índice*. Además, no se pueden utilizar subconsultas en ningún lugar de la *expresión-case* (SQLSTATE 428E4). La expresión case debe devolver un valor entero (SQLSTATE 428E4). Si la *expresión-resultante* devuelve el valor 1, significa que se conserva la entrada de índice; en otro caso se descarta.

Notas

- La creación de una extensión de índice con un nombre de esquema que no existe todavía da como resultado la creación implícita de ese esquema, siempre que el ID de autorización de la sentencia tenga autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN para el esquema se otorga a PUBLIC.

Ejemplos

Ejemplo 1: Este ejemplo crea una extensión de índice llamada *grid_extension*, que utiliza una columna SHAPE de tipo estructurado en una función de tabla llamada *gridEntry* para generar siete claves de índice. Esta extensión de índice proporciona también dos métodos de búsqueda por índice para producir rangos de búsqueda para un argumento de búsqueda determinado.

```
CREATE INDEX EXTENSION GRID_EXTENSION (LEVELS VARCHAR(20) FOR BIT DATA)
FROM SOURCE KEY (SHAPECOL SHAPE)
GENERATE KEY USING GRIDENTRY(SHAPECOL..MBR..XMIN,
                              SHAPECOL..MBR..YMIN,
                              SHAPECOL..MBR..XMAX,
                              SHAPECOL..MBR..YMAX,
                              LEVELS)
WITH TARGET KEY (LEVEL INT, GX INT, GY INT,
                  XMIN INT, YMIN INT, XMAX INT, YMAX INT)
SEARCH METHODS
WHEN SEARCHFIRSTBYSECOND (SEARCHARG SHAPE)
RANGE THROUGH GRIDRANGE(SEARCHARG..MBR..XMIN,
                           SEARCHARG..MBR..YMIN,
                           SEARCHARG..MBR..XMAX,
                           SEARCHARG..MBR..YMAX,
                           LEVELS)
FILTER USING
CASE WHEN (SEARCHARG..MBR..YMIN > YMAX) OR
           (SEARCHARG..MBR..YMAX < YMIN) THEN 0
```

CREATE INDEX EXTENSION

```
ELSE CHECKDUPLICATE(LEVEL, GX, GY,
                    XMIN, YMIN, XMAX, YMAX,
                    SEARCHARG..MBR..XMIN,
                    SEARCHARG..MBR..YMIN,
                    SEARCHARG..MBR..XMAX,
                    SEARCHARG..MBR..YMAX,
                    LEVELS)

END
WHEN SEARCHSECONDBYFIRST (SEARCHARG SHAPE)
RANGE THROUGH GRIDRANGE(SEARCHARG..MBR..XMIN,
                        SEARCHARG..MBR..YMIN,
                        SEARCHARG..MBR..XMAX,
                        SEARCHARG..MBR..YMAX,
                        LEVELS)

FILTER USING
CASE WHEN (SEARCHARG..MBR..YMIN > YMAX) OR
          (SEARCHARG..MBR..YMAX < YMIN) THEN 0
ELSE MBROVERLAP(XMIN, YMIN, XMAX, YMAX,
                SEARCHARG..MBR..XMIN,
                SEARCHARG..MBR..YMIN,
                SEARCHARG..MBR..XMAX,
                SEARCHARG..MBR..YMAX)

END
```

CREATE METHOD

CREATE METHOD

Esta sentencia asocia un cuerpo de método con una especificación de método que ya forma parte de la definición de un tipo estructurado definido por el usuario.

Invocación

Esta sentencia se puede intercalar en un programa de aplicación o se puede emitir utilizando sentencias de SQL dinámicas. Es una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

Autorización

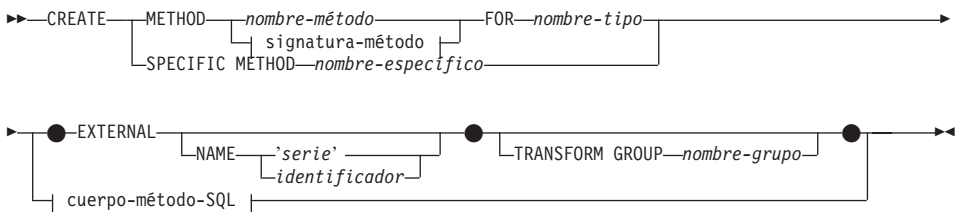
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Privilegio CREATEIN para el esquema del tipo estructurado referenciado en la sentencia CREATE METHOD
- Definidor del tipo estructurado referenciado en la sentencia CREATE METHOD.

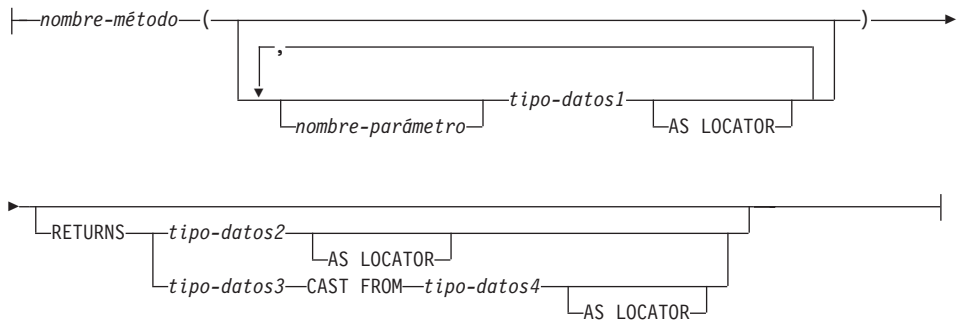
Si el ID de autorización de la sentencia no tiene autorización SYSADM ni DBADM, y el método identifica una tabla o vista en la sentencia RETURN, los privilegios que tiene el ID de autorización de la sentencia (además de los privilegios de grupo) deben incluir SELECT WITH GRANT OPTION para cada tabla y vista identificada.

Si el ID de autorización no tiene la autorización suficiente para realizar la operación, se produce un error (SQLSTATE 42502).

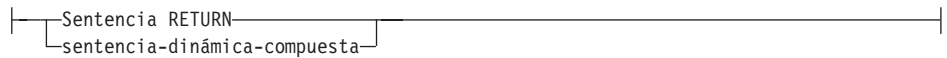
Sintaxis



signatura-método:



cuerpo-método-SQL:



Descripción

METHOD

Identifica una especificación de método existente que está asociada a un tipo estructurado definido por el usuario. La especificación-método se puede identificar de varias maneras:

nombre-método

Designa la especificación de método para la que se define un cuerpo de método. El esquema implícito es el esquema del tipo indicado (*nombre-tipo*). Debe existir una sola especificación de método para *nombre-tipo* que tenga este *nombre-método* (SQLSTATE 42725).

signatura-método

Proporciona la signatura del método, que identifica de manera exclusiva el método que se debe definir. La signatura de método debe coincidir con la especificación proporcionada en la sentencia CREATE TYPE o ALTER TYPE (SQLSTATE 42883).

nombre-método

Designa la especificación de método para la que se define un cuerpo de método. El esquema implícito es el esquema del tipo indicado (*nombre-tipo*).

nombre-parámetro

Designa el nombre del parámetro. Si la signatura de método proporciona nombres de parámetros, deben coincidir exactamente con las partes correspondientes de la especificación de método asociada. Esta sentencia da soporte a los nombres de parámetros sólo con fines de documentación.

CREATE METHOD

tipo-datos1

Especifica el tipo de datos de cada parámetro.

AS LOCATOR

Para los tipos LOB o los tipos diferenciados que se basan en un tipo LOB, se puede añadir la cláusula AS LOCATOR.

RETURNS

Esta cláusula identifica la salida del método. Si la signatura de método proporciona una cláusula RETURNS, ésta debe coincidir exactamente con la parte correspondiente de la especificación de método asociada existente en CREATE TYPE. Esta sentencia da soporte a la cláusula RETURNS sólo con fines de documentación.

tipo-datos2

Especifica el tipo de datos de la salida.

AS LOCATOR

Para tipos LOB o tipos diferenciados que están basados en tipos LOB, se puede añadir la cláusula AS LOCATOR.

Esto indica que el método debe devolver un localizador de LOB, en lugar del valor real.

tipo-datos3 **CAST FROM** *tipo-datos4*

Este formato de cláusula RETURNS se utiliza para devolver un tipo de datos diferente a la sentencia de invocación del tipo de datos que se ha devuelto por el código de función.

AS LOCATOR

Para los tipos LOB o tipos diferenciados basados en un tipo LOB, se puede utilizar la cláusula AS LOCATOR para indicar que el método debe devolver un localizador de LOB, en lugar del valor real.

FOR *nombre-tipo*

Designa el tipo para el cual se debe asociar el método especificado. El nombre debe identificar un tipo que ya esté descrito en el catálogo (SQLSTATE 42704). En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objetos no calificados.

SPECIFIC METHOD *nombre-específico*

Identifica el método concreto, utilizando el nombre específico que se especificó o se tomó por omisión al ejecutar CREATE TYPE. El nombre-específico debe identificar una especificación de método del esquema nombrado o implícito; de lo contrario se produce un error (SQLSTATE 42704).

EXTERNAL

Esta cláusula indica que se utiliza la sentencia CREATE METHOD para registrar un método, basándose en el código escrito en un lenguaje de programación externo y de acuerdo con los convenios documentados para enlaces e interfaces. La especificación-método asociada de CREATE TYPE debe especificar un valor distinto de SQL para LANGUAGE. Cuando se invoca el método, el sujeto del método se pasa a la implementación como primer parámetro implícito.

Si no se especifica la cláusula NAME, se supone "NAME nombre-método".

NAME

Esta cláusula identifica el nombre del código escrito por el usuario que aplica el método que se está definiendo.

'serie'

La opción 'serie' es una constante de tipo serie con un máximo de 254 caracteres. El formato que se utiliza para la serie depende de la opción LANGUAGE especificada. Vea "CREATE FUNCTION (Escalar externa)" en la página 696 para obtener más información sobre los convenios específicos para lenguajes.

identificador

Este identificador especificado es un identificador SQL. El identificador SQL se utiliza como id-biblioteca en la serie. A menos que sea un identificador delimitado, se convierte a mayúsculas. Si el identificador está calificado con un nombre de esquema, no se tiene en cuenta la parte correspondiente al nombre de esquema. Esta modalidad de NAME sólo puede utilizarse con LANGUAGE C (tal como está definido en la especificación-método en CREATE TYPE).

TRANSFORM GROUP *nombre-grupo*

Indica el grupo de transformación que se utiliza, cuando se invoca el método, para las transformaciones de tipo estructurado definidas por el usuario. Es necesaria una transformación, pues la definición de método incluye un tipo estructurado definido por el usuario.

Es muy recomendable especificar un nombre de grupo de transformación; si no se especifica esta cláusula, se utiliza el nombre de grupo por omisión, DB2_FUNCTION. Si el nombre-grupo especificado (o tomado por omisión) no está definido para un tipo estructurado referenciado, se produce en error (SQLSTATE 42741). Similarmente, si una función de transformación necesaria FROM SQL o TO SQL no está definida para el nombre-grupo y tipo estructurado proporcionados, se produce un error (SQLSTATE 42744).

CREATE METHOD

cuerpo-método-SQL

El cuerpo-método-SQL define cómo se implementa el método si la especificación de método en CREATE TYPE es LANGUAGE SQL.

El cuerpo-método-SQL debe adaptarse a las siguientes partes de la especificación de método:

- DETERMINISTIC o NOT DETERMINISTIC (SQLSTATE 428C2)
- EXTERNAL ACTION o NO EXTERNAL ACTION (SQLSTATE 428C2)
- CONTAINS SQL o READS SQL DATA (SQLSTATE 42985)

En el cuerpo-método-SQL se puede hacer referencia a nombres de parámetros. El sujeto del método se pasa a la implementación de método como un primer parámetro implícito llamado SELF.

Para obtener detalles adicionales, consulte el apartado “SQL compuesto (dinámico)” en la página 642 y el apartado “Sentencia RETURN” en la página 1245.

Normas

Para poder utilizar CREATE METHOD, se debe definir previamente la especificación de método utilizando la sentencia CREATE TYPE o ALTER TYPE (SQLSTATE 42723).

Ejemplos

Ejemplo 1:

```
CREATE METHOD BONUS (RATE DOUBLE)
FOR EMP
RETURN SELF..SALARY * RATE
```

Ejemplo 2:

```
CREATE METHOD SAMEZIP (addr address_t)
RETURNS INTEGER
FOR address_t
RETURN
(CASE
  WHEN (self..zip = addr..zip)
  THEN 1
  ELSE 0
END)
```

Ejemplo 3:

```
CREATE METHOD DISTANCE (address_t)
FOR address_t
EXTERNAL NAME 'addresslib!distance'
TRANSFORM GROUP func_group
```

CREATE NICKNAME

La sentencia CREATE NICKNAME crea un apodo para una tabla o una vista de la fuente de datos.

Invocación

Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

Autorización

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Autorización IMPLICIT_SCHEMA para la base de datos federada, si no existe el nombre de esquema implícito o explícito del apodo.
- Privilegio CREATEIN para el esquema, si existe el nombre de esquema del apodo

Además, el ID de autorización del usuario en la fuente de datos debe tener el privilegio para seleccionar en el catálogo de la fuente de datos los metadatos acerca de la tabla o la vista para los que se está creando el apodo.

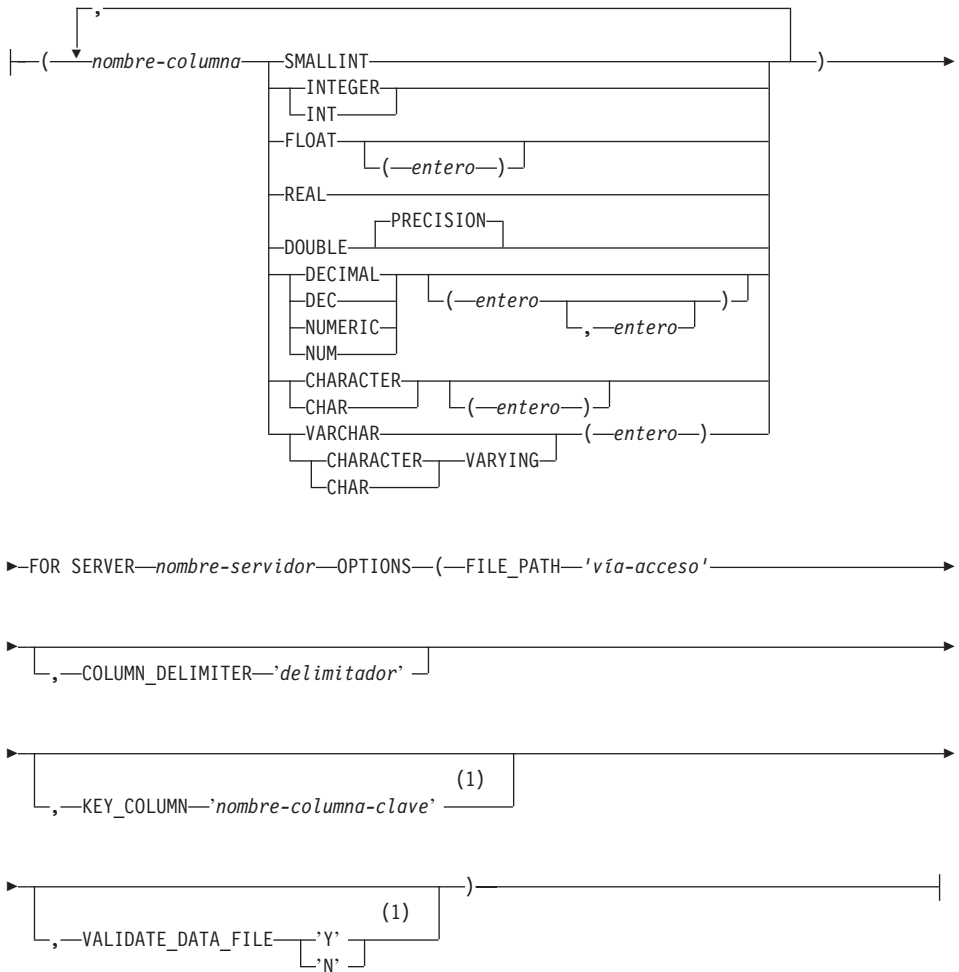
Sintaxis

```

▶▶ CREATE NICKNAME apodo
      └─┬─ FOR nombre-objeto-remoto ─┬─▶
        │                             │
        └─┬─ archivos estructurados-tabla ─┬─▶
  
```

archivos estructurados-tabla:

CREATE NICKNAME



Notas:

- 1 Opcional sólo para archivos clasificados.

Descripción

apodo

Nombra el identificador del servidor federado para la tabla o vista a la que *nombre-objeto-remoto* hace referencia o el archivo estructurado-tabla al que se está accediendo. El apodo, incluido el calificador implícito o explícito, no debe designar una tabla, vista, seudónimo o apodo descritos en el catálogo. El nombre de esquema no debe empezar por SYS (SQLSTATE 42939).

nombre-objeto-remoto

Nombra un identificador formado por tres partes con este formato:

nombre-fuente-datos.nombre-esquema-remoto.nombre-tabla-remota

donde:

nombre-fuente-datos

Nombra la fuente de datos que contiene la tabla o la vista para la cual se está creando el apodo. El *nombre-fuente-datos* es el mismo nombre que se ha asignado a la fuente de datos en la sentencia CREATE SERVER.

nombre-esquema-remoto

Nombra el esquema al que pertenece la tabla o la vista.

nombre-tabla-remota

Nombra uno de los identificadores siguientes:

- El nombre o un seudónimo de una tabla o una vista de la familia DB2
- El nombre de una tabla o una vista Oracle
- El *nombre-tabla* no puede ser una tabla temporal declarada (SQLSTATE 42995)

nombre-columna

Nombre exclusivo que se da a cada campo del archivo estructurado-tabla. Cada *nombre-columna* va seguido de su tipo de datos. Sólo se soportan las columnas de tipo SMALLINT, INTEGER, FLOAT, REAL, DOUBLE, DECIMAL, CHAR y VARCHAR.

SMALLINT

Para especificar un entero pequeño.

INTEGER o INT

Para especificar un entero grande.

FLOAT(entero)

Para especificar un número de coma flotante de precisión simple o doble, en función del valor del entero. El valor del entero debe estar en el rango de 1 a 53. Los valores del 1 al 24 indican precisión simple y los valores del 25 al 53 indican precisión doble.

REAL

Para especificar un valor de coma flotante de precisión simple.

DOUBLE o DOUBLE PRECISION

Para especificar un valor de coma flotante de precisión doble.

FLOAT

Para especificar un valor de coma flotante de precisión doble.

CREATE NICKNAME

DECIMAL(*entero-precisión, entero-escala*) o **DEC**(*entero-precisión, entero-escala*)

Para especificar un número decimal.

El primer entero es la precisión del número; es decir, el número total de dígitos. Este valor puede estar en el rango de 1 a 31. El segundo entero es la escala del número; es decir, el número de dígitos a la derecha de la coma decimal. Este valor puede estar en el rango de 0 a la precisión del número. Si no se especifican la precisión ni la escala, se emplean los valores por omisión 5,0.

Las palabras **NUMERIC** y **NUM** pueden utilizarse como sinónimos de **DECIMAL** y **DEC**.

CHARACTER(*entero*) o **CHAR**(*entero*) o **CHARACTER** o **CHAR**

Para especificar una serie de caracteres de longitud fija de una longitud de *entero*, que puede estar en el rango de 1 a 254. Si se omite la especificación de la longitud, se supone que la longitud es de 1 carácter.

VARCHAR(*entero*) o **CHARACTER VARYING**(*entero*) o **CHAR VARYING**(*entero*)

Para una serie de caracteres de longitud variable de una longitud máxima de *entero*, que puede estar en el rango de 1 a 32672.

nombre-servidor

Identifica la fuente de datos descrita en el catálogo que se utiliza para acceder al archivo estructurado-tabla. Si el archivo está clasificado, el servidor especificado debe ser de tipo **SORTED**; de lo contrario, especifique un servidor de tipo **UNSORTED**.

'vía-acceso'

Vía de acceso totalmente calificada al archivo estructurado-tabla al que se está accediendo. El archivo de datos debe ser un archivo estándar o un enlace simbólico y no puede ser un conducto o un tipo de archivo no estándar. Los archivos de datos tienen que ser legibles por el propietario de la instancia DB2. Para obtener más información sobre los propietarios de instancias, consulte la publicación *Administration Guide*.

'delimitador'

Delimitador para separar columnas del archivo estructurado-tabla. Si no se define ningún delimitador de columna, el delimitador de columna toma por omisión la coma (','). El delimitador de columna no puede existir como datos válidos para una columna. Por ejemplo, no se puede utilizar un delimitador de coma si una de las columnas contiene datos con comas incorporadas.

'nombre-columna-clave'

Nombre de la columna del archivo que se utiliza para formar la clave en la que se clasifica el archivo. Utilice esta opción sólo para archivos clasificados. No es sensible a las mayúsculas y minúsculas.

Sólo se soportan claves de columna-simple. El valor debe ser el nombre de una columna definida en la sentencia CREATE NICKNAME. La columna debe clasificarse en orden ascendente. Si no se especifica el valor para un servidor clasificado, dicho valor toma por omisión la primera columna del archivo con apodo.

VALIDATE_DATA_FILE

Para archivos clasificados, esta opción especifica si el reiniciador verifica que la columna clave esté clasificada en orden ascendente. Los únicos valores válidos para esta opción son 'Y' o 'N'. La comprobación se realiza una vez en el momento del registro. Si no se especifica esta opción, no tiene lugar ninguna validación.

Notas

- La tabla o la vista a la que el apodo hace referencia ya debe existir en la fuente de datos indicada por el primer calificador de *nombre-objeto-remoto*.
- El servidor federado no soporta los tipos de datos de la fuente de datos que corresponden a los siguientes tipos de datos de DB2: LONG VARCHAR, LONG VARGRAPHIC, DATALINK, tipos LOB (gran objeto) y tipos definidos por el usuario. Cuando se define un apodo para una tabla o una vista de fuente de datos, sólo se definirán en la base de datos federada las columnas de la tabla o la vista que tengan tipos de datos soportados y sólo se podrán consultar dichas columnas. Cuando la sentencia CREATE NICKNAME se ejecuta en una tabla o una vista que tiene columnas con tipos de datos no soportados, se emite un error.
- Puesto que los tipos pueden ser incompatibles entre fuentes de datos, el servidor federado realiza pequeños ajustes para almacenar localmente los datos del catálogo remoto, según proceda. Consulte el manual *Application Development Guide* para conocer detalles.
- La longitud máxima permitida de los nombres de índice de DB2 es de 18 caracteres. Si se está creando un apodo para una tabla que tiene un índice cuyo nombre excede esta longitud, no se cataloga todo el nombre. En su lugar, DB2 lo trunca a 18 caracteres. Si la serie formada por estos caracteres no es exclusiva en el esquema al que pertenece el índice, DB2 intenta convertirla en exclusiva sustituyendo el último carácter por un 0. Si el resultado continúa sin ser exclusivo, DB2 cambia el último carácter por un 1. DB2 repite este proceso con los números del 2 al 9 y, si es necesario, con los números del 0 al 9 para el decimoséptimo carácter del nombre, para el decimosexto carácter y así sucesivamente, hasta que se genera un nombre exclusivo. Como ilustración: El índice de la tabla de fuente de datos se llama ABCDEFGHIJKLMNOPQRSTUVWXYZ. Los nombres ABCDEFGHIJKLMNOPQR y ABCDEFGHIJKLMNOPQ0 ya existen en el esquema al que pertenece el índice. El nuevo nombre tiene más de 18 caracteres; por lo tanto, DB2 lo trunca a ABCDEFGHIJKLMNOPQR. Puesto que este nombre ya existe en el esquema, DB2 cambia la versión truncada por ABCDEFGHIJKLMNOPQ0. Puesto que este nombre también existe,

CREATE NICKNAME

DB2 cambia la versión truncada por ABCDEFGHIJKLMNOPQ1. Este nombre aún no existe en el esquema, por lo que DB2 lo acepta como nuevo nombre.

- Cuando se crea un apodo para una tabla o una vista, DB2 almacena los nombres de las columnas de la tabla o la vista en el catálogo. Si un nombre excede la longitud máxima permitida para los nombres de columna de DB2 (30 caracteres), DB2 trunca el nombre hasta esta longitud. Si la versión truncada no es exclusiva entre los demás nombres de las columnas de la tabla o la vista, DB2 lo convierte en exclusivo siguiendo el procedimiento descrito en el párrafo anterior.

Ejemplos

Ejemplo 1: Cree un apodo para una vista, DEPARTMENT, que esté en un esquema llamado HEDGES. Esta vista se almacena en una fuente de datos DB2 Universal Database para OS/390 llamada OS390A.

```
CREATE NICKNAME DEPT FOR OS390A.HEDGES.DEPARTMENT
```

Ejemplo 2: Seleccione todos los registros de la vista para la cual se ha creado el apodo del Ejemplo 1. Debe hacerse referencia a la vista por su apodo. (Sólo puede hacerse referencia por su propio nombre en las sesiones de paso a través.)

```
SELECT * FROM OS390A.HEDGES.DEPARTMENT No válido
```

```
SELECT * FROM DEPT Válido después de crear el apodo DEPT
```

Ejemplo 3: El ejemplo siguiente muestra una sentencia CREATE NICKNAME para el archivo estructurado-tabla DRUGDATA1.TXT.

```
CREATE NICKNAME DRUGDATA1(DCODE INTEGER,DRUG CHAR(20),MANUFACTURER CHAR(20))  
FOR SERVER biochem_lab  
OPTIONS  
(FILE_PATH '/usr/pat/DRUGDATA1.TXT',  
 COLUMN_DELIMITER ',',  
 KEY_COLUMN 'Dcode',  
 VALIDATE_DATA_FILE 'Y')
```

CREATE NODEGROUP

La sentencia CREATE NODEGROUP crea un nuevo grupo de nodos en la base de datos y asigna particiones o nodos al grupo de nodos y registra la definición del grupo de nodos en el catálogo.

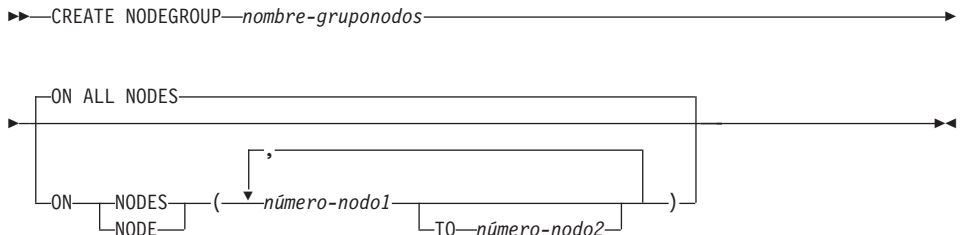
Invocación

Esta sentencia puede incluirse en un programa de aplicación o emitirse de forma interactiva. Se trata de una sentencia ejecutable que se puede preparar dinámicamente. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

Autorización

El ID de autorización de la sentencia debe tener autorización SYSCTRL o SYSADM.

Sintaxis



Descripción

nombre-gruponodos

Indica el nombre del grupo de nodos. Este nombre sólo se compone de una parte. Se trata de un identificador de SQL (ordinario o bien delimitado). El *nombre-gruponodos* no debe identificar un grupo de nodos que ya exista en el catálogo (SQLSTATE 42710). El *nombre-gruponodos* no debe empezar por los caracteres "SYS" ni "IBM" (SQLSTATE 42939).

ON ALL NODES

Especifica que el grupo de nodos se define en todas las particiones definidas en la base de datos (archivo db2nodes.cfg) en el momento en que se crea el grupo de nodos.

Si se añade una partición al sistema de bases de datos, debe emitirse la sentencia ALTER NODEGROUP para incluir esta nueva partición en un grupo de nodos (incluyendo IBMDEFAULTGROUP). Además, debe emitirse el mandato REDISTRIBUTE NODEGROUP para mover los datos a la partición. Consulte los manuales *Administrative API Reference* o *Consulta de mandatos* para obtener más información.

CREATE NODEGROUP

ON NODES

Especifica las particiones específicas que están en nodogrupos. NODE es sinónimo de NODES.

número-nodo1

Especifique un número de partición determinado.⁷³

TO *número-nodo2*

Especifique un rango de números de partición. El valor de *número-nodo2* debe ser mayor o igual que el valor de *número-nodo1* (SQLSTATE 428A9). Todas las particiones entre los números de partición especificados (éstos inclusive) se incluyen en el grupo de nodos.

Normas

- Cada partición o nodo especificado por un número debe estar definido en el archivo `db2nodes.cfg` (SQLSTATE 42729).
- Cada *número-nodo* listado en la cláusula ON NODES debe aparecer como máximo una vez (SQLSTATE 42728).
- Un *número-nodo* válido está entre 0 y 999 inclusive (SQLSTATE 42729).

Notas

- Esta sentencia crea un mapa de particionamiento para el grupo de nodos (Vea “Partición de datos entre múltiples particiones” en la página 40 para obtener más información). Se genera un identificador de mapa de particionamiento (PMAP_ID) para cada mapa de particionamiento. Esta información se anota en el catálogo y se puede recuperar de SYSCAT.NODEGROUPS y SYSCAT.PARTITIONMAPS. Cada entrada del mapa de particionamiento especifica la partición de destino donde residen todas las filas que se generan aleatoriamente. Para un grupo de nodos de partición única, el mapa de particionamiento correspondiente sólo tiene una entrada. Para un grupo de nodos de varias particiones, el mapa de particionamiento correspondiente tiene 4.096 entradas, donde los números de partición se asignan rotatoriamente a las entradas del mapa, de forma predefinida.

Ejemplo

Suponga que tiene una base de datos particionada con seis particiones definidas como: 0, 1, 2, 5, 7 y 8.

- Suponga que desea crear un grupo de nodos llamado MAXGROUP en las seis particiones. La sentencia es la siguiente:

```
CREATE NODEGROUP MAXGROUP  
ON ALL NODES
```

73. Puede especificarse el nombre-nodo en la forma 'NODEnnnnn' para mantener la compatibilidad con la versión anterior.

CREATE NODEGROUP

- Suponga que desea crear un grupo de nodos MEDGROUP en las particiones 0, 1, 2, 5, 8. La sentencia es la siguiente:

```
CREATE NODEGROUP MEDGROUP  
ON NODES (0 TO 2, 5, 8)
```

- Suponga que desea crear un grupo de nodos MINGROUP de una sola partición en la partición (o nodo)7. La sentencia es la siguiente:

```
CREATE NODEGROUP MINGROUP  
ON NODE (7)
```

Nota: La forma en singular de la palabra clave NODES también se acepta.

CREATE PROCEDURE

CREATE PROCEDURE

Esta sentencia se utiliza para registrar un procedimiento almacenado en un servidor de aplicaciones.

Invocación

Esta sentencia se puede intercalar en un programa de aplicación o se puede emitir utilizando sentencias de SQL dinámicas. Es una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

Autorización

Los privilegios del ID de autorización de la sentencia deben incluir como mínimo uno de los siguientes:

- Autorización SYSADM o DBADM
- Autorización IMPLICIT_SCHEMA para la base de datos, si no existe el nombre de esquema implícito ni explícito del procedimiento.
- Autorización de privilegio BINDADD en la base de datos, si el lenguaje para el procedimiento es SQL
- Privilegio CREATEIN para el esquema, si el nombre de esquema del procedimiento hace referencia a un esquema existente.

Para crear un procedimiento almacenado no restringido, los privilegios del ID de autorización de la sentencia deben incluir también uno de los siguientes como mínimo:

- Autorización CREATE_NOT_FENCED para la base de datos
- Autorización SYSADM o DBADM.

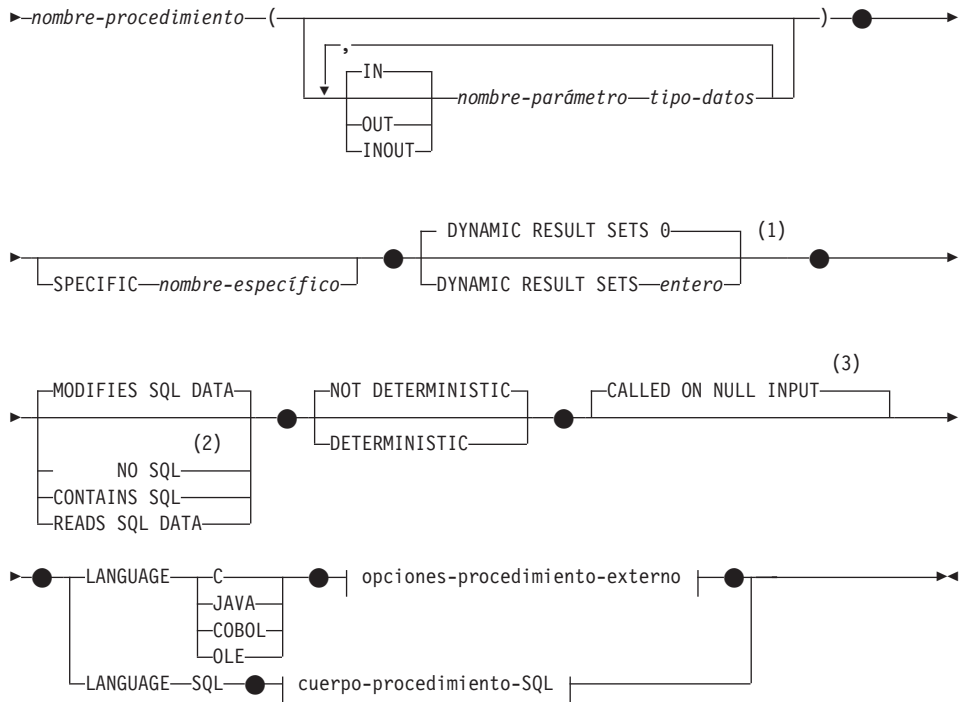
Para crear un procedimiento almacenado restringido, no se necesitan ni autorizaciones ni privilegios adicionales.

Si el ID de autorización no tiene la autorización suficiente para realizar la operación, se produce un error (SQLSTATE 42502).

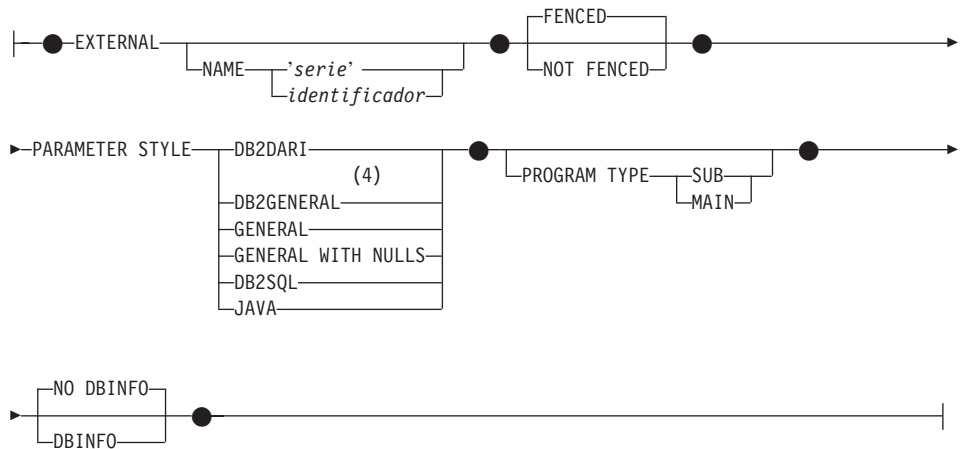
Sintaxis

►—CREATE PROCEDURE—►

CREATE PROCEDURE



opciones-procedimiento-externo:



cuerpo-procedimiento-SQL:

CREATE PROCEDURE

|—sentencia_procedimiento-SQL—|

Notas:

- 1 Puede especificarse RESULT SETS en lugar de DYNAMIC RESULT SETS.
- 2 NO SQL no es una elección válida para LANGUAGE SQL.
- 3 Puede especificarse NULL CALL en lugar de CALLED ON NULL INPUT.
- 4 Puede especificarse DB2GENRL en lugar de DB2GENERAL, puede especificarse SIMPLE CALL en lugar de GENERAL y puede especificarse SIMPLE CALL WITH NULLS en lugar de GENERAL WITH NULLS.

Descripción

nombre-procedimiento

Indica el nombre del procedimiento que se está definiendo. Es un nombre calificado o no calificado que designa un procedimiento. La forma no calificada de *nombre-procedimiento* es un identificador SQL (con una longitud máxima de 128). En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objetos no calificados. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL.

El nombre, incluidos los calificadores implícitos o explícitos, junto con el número de parámetros no debe designar un procedimiento descrito en el catálogo (SQLSTATE 42723). El nombre no calificado, junto con el número de parámetros es, por supuesto, exclusivo en el esquema, pero no necesita ser exclusivo en todos esquemas.

Si se especifica un nombre compuesto de dos partes, el *nombre-esquema* no puede empezar por "SYS". De lo contrario, se genera un error (SQLSTATE 42939).

(IN | OUT | INOUT *nombre-parámetro tipo-datos*,...)

Identifica los parámetros del procedimiento y especifica la modalidad, nombre y tipo de datos de cada parámetro. Debe especificarse una entrada de la lista para cada parámetro que el procedimiento va a esperar.

Es posible registrar un procedimiento que no tenga ningún parámetro. En tal caso, los paréntesis deben seguir codificados, sin tipos de datos intermedios. Por ejemplo,

```
CREATE PROCEDURE SUBWOOFER() ...
```

En un esquema, no se permite que dos procedimientos que se denominen igual tengan exactamente el mismo número de parámetros. Si hay una signatura duplicada se genera un error de SQL (SQLSTATE 42723).

Por ejemplo, si tenemos las sentencias:

```
CREATE PROCEDURE PART (IN NUMBER INT, OUT PART_NAME CHAR(35)) ...
CREATE PROCEDURE PART (IN COST DECIMAL(5,3), OUT COUNT INT) ...
```

la segunda sentencia fallará porque el número de parámetros del procedimiento es igual aunque los tipos de datos no.

IN | OUT | INOUT

Especifica la modalidad del parámetro.

- IN - el parámetro es de sólo entrada
- OUT - el parámetro es de sólo salida
- INOUT - el parámetro es de entrada y de salida

nombre-parámetro

Especifica el nombre del parámetro.

tipo-datos

Especifica el tipo de datos del parámetro.

- Pueden proporcionarse especificaciones y abreviaturas de tipo de datos SQL que pueden especificarse en la definición de *tipo-datos* de la sentencia CREATE TABLE y que tienen una correspondencia en el lenguaje que se utiliza para escribir el procedimiento. Consulte las secciones específicas de los lenguajes del manual *Application Development Guide* para obtener detalles sobre la correlación entre los tipos de datos SQL y los tipos de datos del lenguaje principal con respecto a almacenar procedimientos.
- No se da soporte a los tipos de datos definidos por el usuario (SQLSTATE 42601).

SPECIFIC *nombre-específico*

Proporciona un nombre exclusivo para la instancia del procedimiento que se está definiendo. El nombre específico puede utilizarse al eliminar el procedimiento o realizar un comentario en el procedimiento. No puede utilizarse nunca para invocar el procedimiento. La forma no calificada de *nombre-específico* es un identificador SQL (cuya longitud máxima es 18). La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL. El nombre, incluido el calificador implícito o explícito, no debe identificar otra instancia del procedimiento que exista en el servidor de aplicaciones; de lo contrario se genera un error (SQLSTATE 42710).

El *nombre-específico* puede ser el mismo que un *nombre-procedimiento* ya existente.

CREATE PROCEDURE

Si no se especifica ningún calificador, se emplea el que se haya utilizado para el *nombre-procedimiento*. Si se especifica un calificador, éste debe ser el mismo que el calificador explícito o implícito del *nombre-procedimiento*; de lo contrario, se genera un error (SQLSTATE 42882).

Si no se especifica el *nombre-específico*, el gestor de bases de datos genera un nombre exclusivo. El nombre exclusivo es SQL seguido de una indicación de la hora en caracteres SQLaammddhhmmsshhn.

DYNAMIC RESULT SETS *entero*

Indica el enlace lógico superior estimado de los conjuntos del resultado devueltos para el procedimiento almacenado. Vea “Devolución de conjuntos resultantes de procedimientos almacenados” en el manual *Application Development Guide* para obtener más información.

El valor RESULT SETS puede utilizarse como sinónimo de DYNAMIC RESULT SETS para mantener la compatibilidad con versiones anteriores.

NO SQL, CONTAINS SQL, READS SQL DATA, MODIFIES SQL DATA

Indica si el procedimiento almacenado emite sentencias de SQL y, en caso afirmativo, de qué tipo.

NO SQL

Indica que el procedimiento almacenado no puede ejecutar ninguna sentencia de SQL (SQLSTATE 38001).

CONTAINS SQL

Indica que el procedimiento almacenado puede ejecutar sentencias de SQL que no lean no modifiquen datos SQL (SQLSTATE 38004 ó 42985). Las sentencias que no pueden utilizarse en ningún procedimiento almacenado devuelven un error diferente (SQLSTATE 38003 ó 42985).

READS SQL DATA

Indica que el procedimiento almacenado puede ejecutar algunas sentencias de SQL que no modifiquen datos SQL (SQLSTATE 38002 ó 42985). Las sentencias que no pueden utilizarse en ningún procedimiento almacenado devuelven un error diferente (SQLSTATE 38003 ó 42985).

MODIFIES SQL DATA

Indica que el procedimiento almacenado puede ejecutar cualquier sentencia de SQL excepto aquéllas que no pueden utilizarse en ningún procedimiento almacenado (SQLSTATE 38003 ó 42985).

La tabla siguiente indica si se permite (S) o no (N) que la sentencia de SQL especificada en la primera columna se ejecute en un procedimiento almacenado con la indicación de acceso a datos SQL especificada.

Si se encuentra una sentencia de SQL en un procedimiento almacenado definido con NO SQL, se devuelve el error SQLSTATE 38001. Para otros contextos de ejecución, las sentencias de SQL que no se soportan en ningún contexto devuelven SQLSTATE 38003. Para otras sentencias de SQL no permitidas en un contexto CONTAINS SQL, se devuelve SQLSTATE 38004 y, en un contexto READS SQL DATA, se devuelve SQLSTATE 38002. Durante la creación de un procedimiento SQL, no se permite que se ejecute una sentencia porque la indicación de acceso a datos SQL específica hará que se devuelva SQLSTATE 42985.

Tabla 23. Sentencia de SQL e indicación del acceso a datos SQL

Sentencia de SQL	NO SQL	CONTAINS SQL	READS SQL DATA	MODIFIES SQL DATA
ALTER...	N	N	N	S
BEGIN DECLARE SECTION	S(1)	S	S	S
CALL	N	S(4)	S(4)	S(4)
CLOSE CURSOR	N	N	S	S
COMMENT ON	N	N	N	S
COMMIT	N	N	N	N
COMPOUND SQL	N	S	S	S
CONNECT(2)	N	N	N	N
CREATE	N	N	N	S
DECLARE CURSOR	S(1)	S	S	S
DECLARE GLOBAL TEMPORARY TABLE	N	N	N	S
DELETE	N	N	N	S
DESCRIBE	N	N	S	S
DISCONNECT(2)	N	N	N	N
DROP ...	N	N	N	S
END DECLARE SECTION	S(1)	S	S	S
EXECUTE	N	S(3)	S(3)	S
EXECUTE IMMEDIATE	N	S(3)	S(3)	S
EXPLAIN	N	N	N	S
FETCH	N	N	S	S
FREE LOCATOR	N	S	S	S
FLUSH EVENT MONITOR	N	N	N	S

CREATE PROCEDURE

Tabla 23. Sentencia de SQL e indicación del acceso a datos SQL (continuación)

Sentencia de SQL	NO SQL	CONTAINS SQL	READS SQL DATA	MODIFIES SQL DATA
GRANT ...	N	N	N	S
INCLUDE	S(1)	S	S	S
INSERT	N	N	N	S
LOCK TABLE	N	S	S	S
OPEN CURSOR	N	N	S	S
PREPARE	N	S	S	S
REFRESH TABLE	N	N	N	S
RELEASE CONNECTION(2)	N	N	N	N
RELEASE SAVEPOINT	N	N	N	S
RENAME TABLE	N	N	N	S
REVOKE ...	N	N	N	S
ROLLBACK	N	S	S	S
ROLLBACK TO SAVEPOINT	N	N	N	S
SAVEPOINT	N	N	N	S
SELECT INTO	N	N	S	S
SET CONNECTION(2)	N	N	N	N
SET INTEGRITY	N	N	N	S
SET registro especial	N	S	S	S
UPDATE	N	N	N	S
VALUES INTO	N	N	S	S
WHENEVER	S(1)	S	S	S

Notas:

1. Aunque la opción NO SQL implica que no puede especificarse ninguna sentencia de SQL, las sentencias no ejecutables no están restringidas.
2. Las sentencias de gestión de conexiones no están permitidas en ningún contexto de ejecución de procedimiento almacenado.
3. Depende de la sentencia que se ejecute. La sentencia especificada para la sentencia EXECUTE debe ser una sentencia que esté permitida en el contexto del nivel de acceso SQL que esté vigente. Por ejemplo, si el nivel acceso SQL en vigor es READS SQL DATA, la sentencia no deber ser INSERT, UPDATE ni DELETE.

4. En un procedimiento almacenado, una sentencia CALL sólo puede invocar un procedimiento almacenado escrito en el mismo lenguaje de programación que el procedimiento que emite la llamada.

LANGUAGE

Esta cláusula obligatoria se emplea para especificar el convenio de la interfaz de lenguaje en el que se está escrito el cuerpo del procedimiento almacenado.

C Esto significa que el gestor de bases de datos llamará al procedimiento almacenado como si fuese un procedimiento C. El procedimiento almacenado debe ajustarse al convenio de llamada y enlace del lenguaje C, tal como se define por el prototipo C ANSI estándar.

JAVA Esto significa que el gestor de bases de datos llamará al procedimiento almacenado como si fuera un método de una clase Java.

COBOL

Esto significa que el gestor de bases de datos llamará al procedimiento como si fuera un procedimiento COBOL.

OLE Esto significa que el gestor de bases de datos llamará al procedimiento almacenado como si fuera un método expuesto por un objeto de automatización OLE. El procedimiento almacenado debe ajustarse a los tipos de datos de automatización y al mecanismo de invocación de OLE. Además, el objeto de automatización OLE necesita implementarse como servidor en proceso (DLL). Estas restricciones se describen en el manual OLE Automation Programmer's Reference.

Sólo se da soporte a LANGUAGE OLE para los procedimientos almacenados en DB2 para Sistemas operativos Windows de 32 bits.

SQL El *cuerpo-procedimiento-SQL* especificado incluye las sentencias que definen el proceso del procedimiento almacenado.

EXTERNAL

Esta cláusula indica que la sentencia CREATE PROCEDURE se emplea para registrar un nuevo procedimiento basado en el código escrito en un lenguaje de programación externo y cumple los convenios estipulados para los enlaces e interfaces.

Si no se especifica la cláusula NAME se supone "NAME *nombre-procedimiento*".

NAME 'serie'

Esta cláusula identifica el nombre del código escrito por el usuario que implanta el procedimiento que se está definiendo.

CREATE PROCEDURE

La opción 'serie' es una constante de serie con un máximo de 254 caracteres. El formato que se utiliza para la serie depende de la opción LANGUAGE especificada.

- Para LANGUAGE C:

La *serie* especificada constituye el nombre de la biblioteca y el procedimiento de la biblioteca que el gestor de bases de datos invoca para ejecutar el procedimiento almacenado que se está creando (CREATE). Al ejecutar la sentencia CREATE PROCEDURE, no es preciso que exista la biblioteca (ni el procedimiento dentro de la biblioteca). Sin embargo, cuando se llama el procedimiento, deben existir la biblioteca y el procedimiento en la biblioteca y deben ser accesibles desde la máquina servidora de la base de datos.

→ ' *id_biblioteca* *id_proc* →
└── *id_vía_absoluta* ─┘ └── *id_proc* ─┘

El nombre debe estar acotado por apóstrofes. Dentro de los apóstrofes no puede haber ningún espacio en blanco adicional.

id_biblioteca

Identifica el nombre de la biblioteca donde reside el procedimiento. El gestor de bases de datos buscará la biblioteca en los directorios .../sqllib/function/unfenced y .../sqllib/function (sistemas basados en UNIX), o en los directorios ...*nombre_instancia*\function\unfenced y ...*nombre_instancia*\function (OS/2, Sistemas operativos Windows de 32 bits según lo especificado por la variable de registro DB2INSTPROF), donde el gestor de bases de datos localizará el directorio de control sqllib utilizado para ejecutar el gestor de bases de datos. Por ejemplo, el directorio de control sqllib en los sistemas basados en UNIX es /u/\$DB2INSTANCE/sqllib.

Si 'miproc' es el *id_biblioteca* en un sistema basado en UNIX, el gestor de bases de datos buscará el procedimiento en la biblioteca /u/production/sqllib/function/unfenced/mifunc y /u/production/sqllib/function/mifunc, cuando el gestor de bases de datos se ejecuta desde /u/production.

Para, Sistemas operativos Windows de 32 bits, el gestor de bases de datos buscará en LIBPATH o PATH si el *id_biblioteca* no se encuentra en el directorio function, y el procedimiento se ejecutará como procedimiento restringido.

Los procedimientos almacenados en cualquiera de estos directorios no utilizan ninguno de los atributos registrados.

id_vía_absoluta

Identifica el nombre completo de vía de acceso del procedimiento.

En un sistema basado en UNIX, por ejemplo, `'/u/jchui/milib/miproc'` haría que el gestor de bases de datos buscara en `/u/jchui/milib` el procedimiento `miproc`.

En el OS/2 y Sistemas operativos Windows de 32 bits, si la vía de acceso del procedimiento es `'d:\mibibl\miproc'`, el gestor de bases de datos cargará el archivo `miproc.dll` desde el directorio `d:\mibibl`.

Si se especifica una vía de acceso absoluta, el procedimiento se ejecutará como restringido, sin tener en cuenta el atributo `FENCED / NOT FENCED`.

! id_proc

Identifica el nombre del punto de entrada del procedimiento que se ha de invocar. El signo `!` sirve como delimitador entre el `id` de biblioteca y el `id` de procedimiento. Si se omite `! id_proc`, el gestor de bases de datos utilizará el punto de entrada por omisión establecido al enlazar la biblioteca.

En un sistema basado en UNIX, por ejemplo, `'mimod!proc8'` dirige el gestor de bases de datos para que busque la biblioteca `$inst_home_dir/sqllib/function/mimod` y que utilice el punto de entrada `proc8` dentro de dicha biblioteca.

En OS/2, Sistemas operativos Windows de 32 bits `'mimod!proc8'` dirige al gestor de bases de datos para que cargue el archivo `mimod.dll` y llame al procedimiento `proc8()` de la biblioteca de enlace dinámico (DLL).

Si la composición de la serie de caracteres no es correcta, se produce un error (SQLSTATE 42878).

El cuerpo de cada procedimiento almacenado debe estar en un directorio que está montado y disponible en todas las particiones de la base de datos.

- Para LANGUAGE JAVA:

La *serie* especificada contiene el identificador opcional del archivo `jar`, el identificador de clase y el identificador de método, que el gestor de bases de datos invoca para ejecutar el procedimiento almacenado que se está creando. No es preciso que existan el identificador de clase y el identificador de método cuando se ejecuta la sentencia `CREATE PROCEDURE`. Si se especifica un `id_jar`, éste debe existir cuando se ejecuta la sentencia `CREATE`

CREATE PROCEDURE

PROCEDURE. Sin embargo, cuando se invoca el procedimiento, el identificador de clase y el identificador de método deben existir y ser accesibles desde la máquina del servidor de bases de datos, de lo contrario se produce un error (SQLSTATE 42884).

► ' id_jar ; id_clase . id_método ' ►

El nombre debe estar acotado por apóstrofes. Dentro de los apóstrofes no puede haber ningún espacio en blanco adicional.

id_jar

Designa el identificador de jar que se asignó a la colección jar cuando se instaló en la base de datos. Puede ser un identificador simple o un identificador calificado por un esquema. Algunos ejemplos son 'miJar' y 'miEsquema.miJar'.

id_clase

Identifica el identificador de clase del objeto Java. Si la clase forma parte de un paquete, la parte del identificador de clase debe incluir el prefijo completo del paquete, por ejemplo, 'miPaqs.StoredProcs'. La máquina virtual Java buscará en el directorio './miPaqs/UserFuncs/' las clases. En OS/2 y Sistemas operativos Windows de 32 bits la máquina virtual Java buscará en el directorio './misPaqs\StoredProcs\'.

id_método

Identifica el nombre de método de la clase Java que se ha de invocar.

- Para LANGUAGE OLE:

La serie de caracteres especificada es el identificador de programa OLE (*idprog*) o el identificador de clase (*idcls*), y el identificador de método (*id_método*), que el gestor de bases de datos invoca para ejecutar el procedimiento almacenado creado por la sentencia. No es necesario que el identificador de programa o de clase ni el identificador de método existan cuando se ejecuta la sentencia CREATE PROCEDURE. Sin embargo, cuando el procedimiento se utiliza en una sentencia CALL, el identificador de método debe existir y ser accesible desde la máquina del servidor de bases de datos; de lo contrario se produce un error (SQLSTATE 42724).

► ' idprog ! id_método ' ►
└─ idcls ─┘

El nombre debe estar acotado por apóstrofes. Dentro de los apóstrofes no puede haber ningún espacio en blanco adicional.

idprog

Identifica el identificador de programa del objeto OLE.

El gestor de bases de datos no interpreta los *idprog*, sino que sólo se reenvían al controlador de automatización OLE durante la ejecución. El objeto OLE especificado debe poderse crear y debe dar soporte al enlace tardío (llamado también enlace basado en IDispatch). Por convenio, los *idprog* tienen el formato siguiente:

<nombre_programa>.<nombre_componente>.<versión>

Puesto que sólo es un convenio, y no una norma, los *idprog* pueden tener en realidad un formato diferente.

idcls

Designa el identificador de clase del objeto OLE que se debe crear. Puede utilizarse como una alternativa a la especificación de un *idprog* cuando un objeto OLE no está registrado con un *idprog*. El *idcls* tiene este formato:

{nnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnnn}

donde 'n' es un carácter alfanumérico. El gestor de bases de datos no interpreta los *idcls*, sino que sólo se reenvían a las API de OLE durante la ejecución.

id_método

Identifica el nombre de método del objeto OLE que se ha de invocar.

NAME *identificador*

Este *identificador* especificado es un identificador SQL. El identificador SQL se utiliza como el *id-biblioteca* en la serie. A menos que sea un identificador delimitado, se convierte a mayúsculas. Si el identificador está calificado con un nombre de esquema, no se tiene en cuenta la parte correspondiente al nombre de esquema. Este formato de NAME sólo puede utilizarse con LANGUAGE C.

FENCED o **NOT FENCED**

Esta cláusula especifica si el procedimiento almacenado se considera "seguro" (NOT FENCED) o no (FENCED) para ejecutarse en el proceso o espacio de direcciones del entorno operativo del gestor de bases de datos.

Si un procedimiento almacenado se registra como FENCED, el gestor de bases de datos aísla los recursos internos de base de datos (por ejemplo, almacenamientos intermedios de datos) para que el procedimiento no pueda acceder a ellos. Todos los procedimientos tienen la opción de ejecutar como FENCED o NOT FENCED. En

CREATE PROCEDURE

general, un procedimiento que se ejecute como FENCED no funcionará tan bien como otro de iguales características que se ejecute como NOT FENCED.

Si el procedimiento almacenado está ubicado en el directorio `.../sqllib/function/unfenced`, en el directorio `.../sqllib/function` (basado en sistemas UNIX) o en el directorio `...\nombre_instancia\function\unfenced` y en el directorio `...\nombre_instancia\function` (OS/2, Sistemas operativos Windows de 32 bits), entonces el atributo registrado FENCED o NOT FENCED (y cualquier otro atributo registrado) serán ignorados.

Nota: La utilización de NOT FENCED para procedimientos no verificados de manera adecuada puede comprometer la integridad de DB2. DB2 toma algunas precauciones contra muchas de las anomalías más habituales debido a descuidos, pero no puede garantizar la completa integridad de los datos cuando se utilizan procedimientos almacenados NOT FENCED.

Para cambiar FENCED por NOT FENCED, es preciso volver a registrar el procedimiento (eliminándolo y después volviéndolo a crear). Para que un procedimiento almacenado se pueda registrar como NOT FENCED es necesaria la autorización SYSADM, la autorización DBADM o una autorización especial (CREATE_NOT_FENCED). Si el procedimiento almacenado se ha definido con LANGUAGE OLE, sólo puede especificarse FENCED .

PARAMETER STYLE

Esta cláusula sirve para especificar los convenios que se emplean para pasar los parámetros a los procedimientos almacenados y para devolver el valor de los mismos.

DB2DARI

Esto significa que el procedimiento almacenado utilizará un parámetro para pasar un convenio que se ajusta a los convenios de enlace y de llamada del lenguaje C. Esto sólo puede especificarse cuando se utiliza LANGUAGE C.

DB2GENERAL

Esto significa que el procedimiento almacenado utilizará un parámetro que pasa un convenio que se define para utilizarlo con métodos Java. Esto sólo puede especificarse cuando se utiliza LANGUAGE JAVA.

El valor DB2GENRL puede utilizarse como sinónimo de DB2GENERAL.

GENERAL

Esto significa que el procedimiento almacenado utilizará un

mecanismo que pasa un mecanismo donde el procedimiento almacenado recibe los parámetros especificados en CALL. Los parámetros se pasan directamente tal como espera el lenguaje. No se utiliza la estructura SQLDA. Esto sólo puede especificarse cuando se utiliza LANGUAGE C o COBOL.

Los indicadores nulos NO se pasan directamente al programa.

El valor SIMPLE CALL puede utilizarse como sinónimo de GENERAL.

GENERAL WITH NULLS

Además de los parámetros en la sentencia CALL tal como se especifica en GENERAL, se pasa otro argumento al procedimiento almacenado. Este argumento adicional contiene un vector de indicadores nulos para cada parámetro en la sentencia CALL. En C, esto sería una matriz de short ints. Esto sólo puede especificarse cuando se utiliza LANGUAGE C o COBOL.

El valor SIMPLE CALL WITH NULLS puede utilizarse como sinónimo de GENERAL.

DB2SQL

Además de los parámetros en la sentencia CALL, se pasan los argumentos siguientes al procedimiento almacenado:

- un indicador NULL para cada parámetro en la sentencia CALL
- el SQLSTATE que se tiene que devolver a DB2
- el nombre calificado del procedimiento almacenado
- el nombre específico del procedimiento almacenado
- la serie de diagnóstico SQL que se tiene que devolver a DB2

Esto sólo puede especificarse cuando se utiliza LANGUAGE C, COBOL u OLE.

JAVA Esto significa que el procedimiento almacenado utilizará un parámetro que pasa un convenio que se ajusta al lenguaje Java y a la especificación de rutinas SQLJ. Los parámetros IN/OUT y OUT se pasarán como matrices de entrada para facilitar los valores de retorno. Esto sólo puede especificarse cuando se utiliza LANGUAGE JAVA.

Los procedimientos PARAMETER STYLE JAVA no soportan las cláusulas DBINFO o PROGRAM TYPE.

Consulte el manual *Application Development Guide* para ver los detalles sobre cómo pasar parámetros.

CREATE PROCEDURE

PROGRAM TYPE

Especifica si el procedimiento almacenado espera parámetros del tipo de una rutina principal o una subrutina.

SUB

El procedimiento almacenado espera que los parámetros se pasen como argumentos separados.

MAIN

El procedimiento almacenado espera que los parámetros se pasen como un contador de argumentos y como un vector de argumentos (argc, argv). El nombre del procedimiento almacenado que se debe invocar también debe ser "main". Los procedimientos almacenados de este tipo deben todavía crearse del mismo modo que una biblioteca compartida, a diferencia de un ejecutable autónomo.

El valor por omisión de PROGRAM TYPE es SUB. PROGRAM TYPE MAIN sólo es válido para LANGUAGE C o COBOL y PARAMETER STYLE GENERAL, GENERAL WITH NULLS o DB2SQL.

DETERMINISTIC o NOT DETERMINISTIC

Esta cláusula especifica si el procedimiento devuelve siempre los mismos resultados para unos valores argumento determinados (DETERMINISTIC) o si el procedimiento depende de algunos valores de estado que afectan a los resultados (NOT DETERMINISTIC). Es decir, un procedimiento DETERMINISTIC debe siempre devolver el mismo resultado ante invocaciones sucesivas con entradas de datos idénticas.

Actualmente esta cláusula no afecta al proceso del procedimiento almacenado.

CALLED ON NULL INPUT

>CALLED ON NULL INPUT siempre se aplica a procedimientos almacenados. Esto significa que se llama al procedimiento almacenado sin tener en cuenta si algún argumento es nulo. Puede devolver un valor nulo o un valor normal (no nulo). Corresponde al procedimiento almacenado comprobar si hay valores argumento nulos.

El valor NULL CALL puede utilizarse como sinónimo de CALLED ON NULL INPUT para mantener la compatibilidad con versiones anteriores.

NO DBINFO o DBINFO

Especifica si se pasa información específica conocida por DB2 al procedimiento almacenado cuando se invoca como argumento en tiempo de una invocación adicional (DBINFO) o no (NO DBINFO). NO DBINFO es el valor por omisión. DBINFO no puede utilizarse

para LANGUAGE OLE (SQLSTATE 42613). Tampoco puede utilizarse para PARAMETER STYLE JAVA, DB2GENERAL ni DB2DARI.

Si se especifica DBINFO, entonces se pasa una estructura al procedimiento almacenado que contiene la información siguiente:

- Nombre de base de datos - el nombre de la base de datos conectada actualmente.
- ID de aplicación - ID de aplicación exclusivo que se establece para cada conexión con la base de datos.
- ID de autorización de aplicación - el ID de autorización en tiempo de ejecución de la aplicación.
- Página de códigos - identifica la página de códigos de la base de datos.
- Nombre de esquema - no se puede aplicar a procedimientos almacenados.
- Nombre de tabla - no se puede aplicar a procedimientos almacenados.
- Nombre de columna - no se puede aplicar a procedimientos almacenados.
- Versión/release de base de datos - identifica la versión, release y nivel de modificación del servidor de la base de datos que invoca al procedimiento almacenado.
- Plataforma - contiene el tipo de plataforma del servidor.
- Números de columnas resultantes de la función de tabla - no es aplicable a los procedimientos almacenados.

Por favor, consulte el manual *Application Development Guide* para obtener información detallada sobre la estructura y sobre cómo ésta pasa al procedimiento almacenado.

cuerpo-procedimiento-SQL

Especifica la sentencia de SQL que forma el cuerpo del procedimiento SQL. Se pueden especificar varias sentencias de procedimiento SQL dentro de una sentencia compuesta. Vea “Capítulo 7. Sentencias de control de SQL” en la página 1209 para obtener más información.

Notas

- Para obtener información sobre la creación de programas para un procedimiento almacenado consulte el manual *Application Development Guide*.
- La creación de un procedimiento con un nombre de esquema que todavía no existe dará como resultado la creación implícita de dicho esquema siempre que el ID de autorización tenga la autorización

CREATE PROCEDURE

IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN en el esquema se otorga a PUBLIC.

- Los valores de los registros especiales utilizados por el llamador son heredados por el procedimiento almacenado durante la invocación y se restauran al devolver el control al llamador. Los registros especiales se pueden modificar dentro de un procedimiento almacenado, pero estos cambios no se aplican al llamador. Esto no es cierto para los procedimientos almacenados preexistentes (aquellos definidos con el estilo de parámetro DB2DARI o situados en la biblioteca por omisión), donde los cambios hechos en los registros especiales de un procedimiento se convierten en los valores del llamador.

Ejemplos

Ejemplo 1: Cree la definición de procedimiento para un procedimiento almacenado, escrito en Java, al que se pasa el número de una pieza y devuelve el coste de la pieza y la cantidad que hay disponible en este momento.

```
CREATE PROCEDURE PARTS_ON_HAND (IN PARTNUM INTEGER,  
                                OUT COST DECIMAL(7,2),  
                                OUT QUANTITY INTEGER)  
                                EXTERNAL NAME 'parts.onhand'  
LANGUAGE JAVA PARAMETER STYLE JAVA
```

Ejemplo 2: Cree la definición de procedimiento para un procedimiento almacenado, escrito en C, al que se pasa el número de un ensamblaje y devuelve el número de las piezas que lo componen, el coste total de las piezas y un conjunto resultante que lista los números de pieza, la cantidad y el coste por unidad de cada pieza.

```
CREATE PROCEDURE ASSEMBLY_PARTS (IN ASSEMBLY_NUM INTEGER,  
                                OUT NUM_PARTS INTEGER,  
                                OUT COST DOUBLE)  
                                EXTERNAL NAME 'piezas!ensamblaje'  
DYNAMIC RESULT SETS 1 NOT FENCED  
LANGUAGE C PARAMETER STYLE GENERAL
```

Ejemplo 3: Creación de un procedimiento SQL que devuelve el salario medio de una plantilla de trabajadores. Se obtendrá un conjunto resultante que contiene el nombre, el puesto de trabajo y el salario de todos los empleados que ganan más que el salario medio.

```
CREATE PROCEDURE MEDIAN_RESULT_SET  
(OUT medianSalary DOUBLE)  
  RESULT SETS 1  
  LANGUAGE SQL  
BEGIN  
  DECLARE v_numRecords INT DEFAULT 1;  
  DECLARE v_counter INT DEFAULT 0;  
  
  DECLARE c1 CURSOR FOR
```

```
        SELECT CAST(salary AS DOUBLE)
            FROM staff
            ORDER BY salary;
    DECLARE c2 CURSOR WITH RETURN FOR
        SELECT name, job, CAST(salary AS INTEGER)
            FROM staff
            WHERE salary > medianSalary
            ORDER BY salary;
    DECLARE EXIT HANDLER FOR NOT FOUND
        SET medianSalary = 6666;
    SET medianSalary = 0;
    SELECT COUNT(*) INTO v_numRecords
        FROM STAFF;
    OPEN c1;
    WHILE v_counter < (v_numRecords / 2 + 1)
        DO FETCH c1 INTO medianSalary;
        SET v_counter = v_counter + 1;
    END WHILE;
    CLOSE c1;
    OPEN c2;
END
```

CREATE SCHEMA

CREATE SCHEMA

La sentencia CREATE SCHEMA define un esquema. También es posible crear algunos objetos y otorgar privilegios en los objetos dentro de la sentencia.

Invocación

Esta sentencia se puede intercalar en un programa de aplicación o se puede emitir utilizando sentencias de SQL dinámicas. Es una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

Autorización

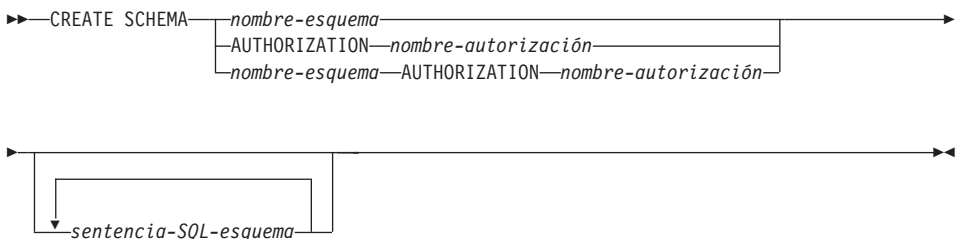
Un ID de autorización que tenga la autorización SYSADM o DBADM puede crear un esquema con cualquier *nombre-esquema* o *nombre-autorización* válido.

Un ID de autorización que no tenga la autorización SYSADM o DBADM sólo puede crear un esquema con un *nombre-esquema* o *nombre-autorización* que coincida con el ID de autorización de la sentencia.

Si la sentencia incluye cualquier *sentencia-SQL-esquema* los privilegios que tiene el *nombre-autorización* (si no se especifica, toma por omisión el ID de autorización de la sentencia) debe incluir como mínimo uno de los siguientes:

- Los privilegios necesarios para ejecutar cada *sentencia-SQL-esquema*
- Autorización SYSADM o DBADM.

Sintaxis



Descripción

nombre-esquema

Indica el nombre del esquema. El nombre no debe identificar un esquema que ya esté descrito en el catálogo (SQLSTATE 42710). El nombre no puede empezar por "SYS" (SQLSTATE 42939). El propietario del esquema es el ID de autorización que ha emitido la sentencia.

AUTHORIZATION *nombre-autorización*

Identifica el usuario que es el propietario del esquema. El valor del

nombre-autorización también se utiliza para nombrar el esquema. El *nombre-autorización* no debe identificar un esquema que ya esté descrito en el catálogo (SQLSTATE 42710).

nombre-esquema **AUTHORIZATION** *nombre-autorización*

Identifica un esquema llamado *nombre-esquema* con el usuario llamado *nombre-autorización* como el propietario del esquema. El *nombre-esquema* no debe identificar un nombre-esquema para un esquema que ya esté descrito en el catálogo (SQLSTATE 42710). El *nombre-esquema* no puede empezar por "SYS" (SQLSTATE 42939).

sentencia-SQL-esquema

Las sentencias de SQL que se pueden incluir como parte de la sentencia CREATE SCHEMA son:

- La sentencia CREATE TABLE con exclusión de las tablas con tipo y las tablas de resumen (vea "CREATE TABLE" en la página 835)
- Sentencia CREATE VIEW excluidas las vistas con tipo (vea "CREATE VIEW" en la página 957)
- La sentencia CREATE INDEX (vea "CREATE INDEX" en la página 774)
- La sentencia COMMENT ON (vea "COMMENT" en la página 628)
- La sentencia GRANT (vea "GRANT (privilegios de apodo, vista o tabla)" en la página 1070).

Notas

- El propietario del esquema se determina de la manera siguiente:
 - Si se especifica la cláusula AUTHORIZATION, el *nombre-autorización* especificado es el propietario del esquema
 - Si no se especifica la cláusula AUTHORIZATION, el ID de autorización que emite la sentencia CREATE SCHEMA es el propietario del esquema.
- Se supone que el propietario del esquema es un usuario (no un grupo).
- Cuando se crea explícitamente el esquema con la sentencia CREATE SCHEMA, se otorga al propietario del esquema los privilegios CREATEIN, DROPIN y ALTERIN en el esquema con la posibilidad de otorgar estos privilegios a otros usuarios.
- La persona que define cualquier objeto creado como parte de la sentencia CREATE SCHEMA es el propietario del esquema. El propietario del esquema es el otorgante de cualquier privilegio que se otorga como parte de la sentencia CREATE SCHEMA.
- Los nombres de objeto no calificados en ninguna sentencia de SQL dentro de la sentencia CREATE SCHEMA se califican implícitamente por el nombre del esquema creado.
- Si la sentencia CREATE contiene un nombre calificado para el objeto que se está creando, el nombre de esquema especificado en el nombre calificado debe ser el mismo que el nombre del esquema que se está creando

CREATE SCHEMA

(SQLSTATE 42875). Cualquier otro objeto que se haga referencia en las sentencias pueden calificarse con un nombre de esquema válido.

- Si se especifica la cláusula `AUTHORIZATION` y se utiliza la autenticación DCE, la pertenencia al grupo del *nombre-autorización* especificada no se tendrá en cuenta en la evaluación de las autorizaciones necesarias para ejecutar las sentencias que siguen a la cláusula. Si el *nombre-autorización* especificado es diferente que el ID de autorización que crea el esquema, puede dar como resultado una anomalía durante la ejecución de la sentencia `CREATE SCHEMA`.
- Es recomendable no utilizar `"SESSION"` como nombre de esquema. Debido a que las tablas temporales declaradas deben estar calificadas por `"SESSION"`, es posible que una aplicación declare una tabla temporal que tenga el mismo nombre que el de una tabla permanente. Una tabla incluida en una sentencia de SQL y que tiene el nombre de esquema `"SESSION"` se convierte (al compilarse la sentencia) en la tabla temporal declarada, y no en la tabla permanente del mismo nombre. Debido a que las sentencias estáticas y dinámicas del SQL intercalado se compilan en momentos diferentes, los resultados dependen del momento en que se define la tabla temporal declarada. Estas cuestiones no se plantean cuando no se utiliza el nombre de esquema `"SESSION"` para definir una tabla permanente, vista oseudónimo.

Ejemplos

Ejemplo 1: Como usuario con autorización `DBADM`, cree un esquema llamado `RICK` con el usuario `RICK` como el propietario.

```
CREATE SCHEMA RICK AUTHORIZATION RICK
```

Ejemplo 2: Cree un esquema que tenga una tabla de piezas del inventario y un índice de los números de piezas. Otorgue la autorización sobre la tabla al usuario `JONES`.

```
CREATE SCHEMA INVENTORY
```

```
CREATE TABLE PART (PARTNO SMALLINT NOT NULL,  
DESCR VARCHAR(24),  
QUANTITY INTEGER)
```

```
CREATE INDEX PARTIND ON PART (PARTNO)
```

```
GRANT ALL ON PART TO JONES
```

Ejemplo 3: Cree un esquema llamado `PERS` con dos tablas que cada una tenga una clave foránea que haga referencia a otra tabla. Es un ejemplo de una característica de la sentencia `CREATE SCHEMA` que permite la creación de un par de tablas como éstas sin la utilización de la sentencia `ALTER TABLE`.

CREATE SCHEMA PERS

```
CREATE TABLE ORG (DEPTNUMB SMALLINT NOT NULL,  
DEPTNAME VARCHAR(14),  
MANAGER SMALLINT,  
DIVISION VARCHAR(10),  
LOCATION VARCHAR(13),  
CONSTRAINT PKEYDNO  
PRIMARY KEY (DEPTNUMB),  
CONSTRAINT FKEYMGR  
FOREIGN KEY (MANAGER)  
REFERENCES STAFF (ID) )
```

```
CREATE TABLE STAFF (ID SMALLINT NOT NULL,  
NAME VARCHAR(9),  
DEPT SMALLINT,  
JOB VARCHAR(5),  
YEARS SMALLINT,  
SALARY DECIMAL(7,2),  
COMM DECIMAL(7,2),  
CONSTRAINT PKEYID  
PRIMARY KEY (ID),  
CONSTRAINT FKEYDNO  
FOREIGN KEY (DEPT)  
REFERENCES ORG (DEPTNUMB) )
```

CREATE SEQUENCE

CREATE SEQUENCE

La sentencia CREATE SEQUENCE crea una secuencia en el servidor de aplicaciones.

Invocación

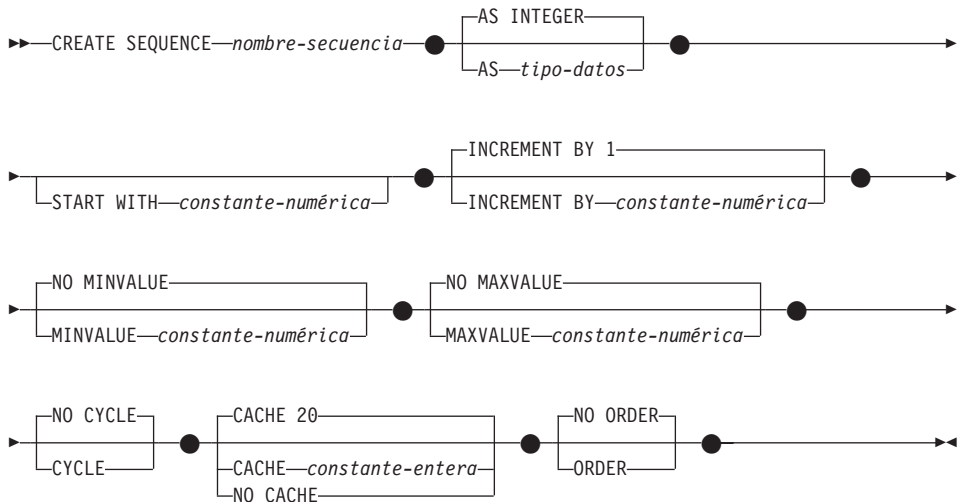
Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse dinámicamente. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

Autorización

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Privilegio CREATEIN para el esquema especificado implícita o explícitamente
- Autorización SYSADM o DBADM

Sintaxis



Descripción

nombre-secuencia

Indica el nombre de la secuencia. La combinación del nombre y del nombre de esquema implícito o explícito no debe identificar una secuencia existente en el servidor actual (SQLSTATE 42710).

El formato no calificado de un nombre-secuencia es un identificador SQL. El formato calificado es un calificador seguido de un punto y un identificador SQL. El calificador es un nombre de esquema.

Si el nombre de secuencia se califica explícitamente con un nombre de esquema, el nombre de esquema no puede empezar con 'SYS' o se producirá un error (SQLSTATE 42939).

AS *tipo-datos*

Especifica el tipo de datos que se debe utilizar para el valor de secuencia. El tipo de datos puede ser cualquier tipo numérico exacto (SMALLINT, INTEGER, BIGINT o DECIMAL) con una escala de cero o un tipo diferenciado o tipo de referencia definido por el usuario para el cual el tipo fuente sea un tipo numérico exacto con una escala de cero (SQLSTATE 42815). El valor por omisión es INTEGER.

START WITH *constante-numérica*

Especifica el primer valor de la secuencia. Este valor puede ser cualquier valor positivo o negativo que pueda asignarse a una columna del tipo de datos asociado con la secuencia (SQLSTATE 42820) y que no tenga dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA). El valor por omisión es MINVALUE para secuencias ascendentes y MAXVALUE para secuencias descendentes.

Este valor no es necesariamente el valor en el que una secuencia realizaría un ciclo después de alcanzar el valor máximo o mínimo de la secuencia. Se puede utilizar la cláusula START WITH para iniciar una secuencia fuera del rango que se usa para los ciclos. El rango utilizado para los ciclos se define mediante MINVALUE y MAXVALUE.

INCREMENT BY *constante-numérica*

Especifica el intervalo entre valores consecutivos de la secuencia. Este valor puede ser cualquier valor positivo o negativo que pueda asignarse a una columna del tipo de datos asociado con la secuencia (SQLSTATE 42820), que no exceda el valor de una constante entera grande (SQLSTATE 42815) y que no tenga dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA).

Si este valor es negativo, se trata de una secuencia descendente. Si este valor es 0 o el valor absoluto es mayor que MAXVALUE - MINVALUE o positivo, se trata de una secuencia ascendente. El valor por omisión es 1.

MINVALUE o NO MINVALUE

Especifica el valor mínimo en el que una secuencia descendente pasa por un ciclo o deja de generar valores o en el que una secuencia ascendente pasa por un ciclo después de alcanzar el valor máximo.

MINVALUE *constante-numérica*

Especifica la constante numérica que es el valor mínimo. Este valor puede ser cualquier valor positivo o negativo que pueda asignarse a

CREATE SEQUENCE

una columna del tipo de datos asociado con la secuencia (SQLSTATE 42820), que no tenga dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA) pero que sea inferior o igual al valor máximo (SQLSTATE 42815).

NO MINVALUE

Para una secuencia ascendente, el valor es el valor de inicio original. Para una secuencia descendente, el valor es el valor mínimo del tipo de datos asociado con la secuencia. Éste es el valor por omisión.

MAXVALUE o NO MAXVALUE

Especifica el valor máximo en el que una secuencia ascendente pasa por un ciclo o deja de generar valores o en el que una secuencia descendente pasa por un ciclo después de alcanzar el valor mínimo.

MAXVALUE *constante-numérica*

Especifica la constante numérica que es el valor máximo. Este valor puede ser cualquier valor positivo o negativo que pueda asignarse a una columna del tipo de datos asociado con la secuencia (SQLSTATE 42820), que no tenga dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA) pero que sea superior o igual al valor mínimo (SQLSTATE 42815).

NO MAXVALUE

Para una secuencia ascendente, el valor es el valor máximo del tipo de datos asociado con la secuencia. Para una secuencia descendente, el valor es el valor de inicio original. Éste es el valor por omisión.

CYCLE o NO CYCLE

Especifica si la secuencia debe continuar generando valores después de alcanzar el valor máximo o el valor mínimo. El límite de la secuencia puede alcanzarse con el siguiente valor que coincide exactamente con la condición de límite o excediendo el valor.

CYCLE

Especifica que se continúan generando valores para esta secuencia después de haber alcanzado el valor máximo o mínimo. Si se utiliza esta opción, cuando una secuencia ascendente haya alcanzado su valor máximo, generará su valor mínimo o cuando una secuencia descendente haya alcanzado su valor mínimo, generará su valor máximo. Los valores máximo y mínimo para la secuencia determinan el rango que se utiliza para el ciclo.

Cuando CYCLE está en vigor, se pueden generar valores duplicados para la secuencia.

NO CYCLE

Especifica que no se generarán valores para la secuencia una vez que se haya alcanzado el valor máximo o mínimo para la secuencia. Éste es el valor por omisión.

CACHE o NO CACHE

Especifica si se deben mantener algunos valores preasignados en memoria para obtener un acceso más rápido. Esta opción se utiliza para el rendimiento y el ajuste.

CACHE *constante-entera*

Especifica el número máximo de valores de secuencia que se preasignan y se mantienen en memoria. La preasignación y el almacenamiento de valores en la antememoria reduce la E/S síncrona en las anotaciones cronológicas cuando se generan valores para la secuencia.

En el caso de producirse una anomalía del sistema, todos los valores de secuencia almacenados en antememoria que no se han utilizando en sentencias confirmadas se pierden (es decir, no se utilizarán nunca). El valor especificado para la opción CACHE es el número máximo de valores de secuencia que puede perderse en caso de anomalía del sistema.

El valor mínimo es 2 (SQLSTATE 42815). El valor por omisión es CACHE 20.

NO CACHE

Especifica que los valores de la secuencia no se deben preasignar. Asegura que no haya ninguna pérdida de valores en el caso de producirse una anomalía del sistema, un cierre o una desactivación de la base de datos. Cuando se especifica esta opción, los valores de la secuencia no se almacenan en la antememoria. En este caso, cada petición de un valor nuevo para la secuencia produce E/S síncrona en las anotaciones cronológicas.

NO ORDER o ORDER

Especifica si los números de secuencia deben generarse según el orden de petición.

ORDER

Especifica que los números de secuencia se generan según el orden de petición.

NO ORDER

Especifica que los números de secuencia no necesitan generarse según el orden de petición. Éste es el valor por omisión.

Notas

- Es posible definir una secuencia constante, es decir, una que devuelva siempre un valor constante. Esto puede realizarse especificando el mismo valor para MINVALUE o MAXVALUE o especificando un valor de INCREMENT de cero. Una secuencia constante puede utilizarse como variable global numérica. ALTER SEQUENCE puede utilizarse para ajustar los valores que se generarán para una secuencia constante.

CREATE SEQUENCE

- Se pueden establecer los ciclos de una secuencia manualmente, utilizando la sentencia ALTER SEQUENCE. Si se especifica NO CYCLE de forma implícita o explícita, se puede reiniciar o ampliar la secuencia utilizando la sentencia ALTER SEQUENCE para hacer que los valores continúen generándose una vez que se haya alcanzado el valor máximo o mínimo para la secuencia.
- El almacenamiento en antememoria de los números de secuencia implica que un rango de números de secuencia puede mantenerse en la memoria para acceder a ellos rápidamente. Cuando una aplicación accede a una secuencia que puede asignar el siguiente número de secuencia desde la antememoria, la asignación de números de secuencia se produce rápidamente. Sin embargo, si una aplicación accede a una secuencia que no puede asignar el siguiente número de secuencia desde la antememoria, puede que para asignar el número de secuencia sea necesario esperar a que se realicen las operaciones de E/S en el almacenamiento permanente. La elección del valor para CACHE debe realizarse teniendo en cuenta los cambios de requisitos del rendimiento y de la aplicación.
- El propietario tiene los privilegios ALTER y USAGE en la nueva secuencia. El propietario sólo puede otorgar el privilegio USAGE y únicamente en PUBLIC.
- También se soporta la sintaxis siguiente: NOMINVALUE, NOMAXVALUE, NOCYCLE, NOCACHE y NOORDER.

Ejemplos

Ejemplo 1: Cree una secuencia llamada ORG_SEQ que empieza en 1, se incrementa en 1, no realiza ciclos y almacena en antememoria 24 valores a la vez:

```
CREATE SEQUENCE ORG_SEQ
  START WITH 1
  INCREMENT BY 1
  NO MAXVALUE
  NO CYCLE
  CACHE 24
```

CREATE SERVER

La sentencia CREATE SERVER ⁷⁴ define una fuente de datos en una base de datos federada.

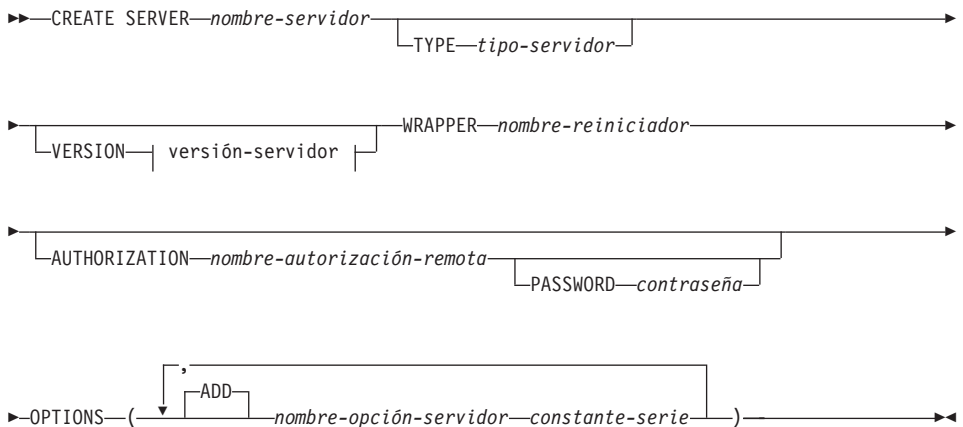
Invocación

Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

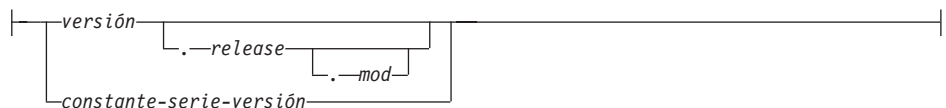
Autorización

El ID de autorización de la sentencia debe tener la autorización SYSADM o DBADM en la base de datos federada.

Sintaxis



versión-servidor:



74. En esta sentencia, el término SERVER y los nombres de parámetros que empiezan por *servidor*- sólo hacen referencia a las fuentes de datos de un sistema federado. No hacen referencia al servidor federado de dicho sistema ni a los servidores de aplicaciones DRDA. Para obtener información acerca de los sistemas federados, consulte "Sistemas federados DB2" en la página 56. Para obtener información acerca de los servidores de aplicaciones DRDA, consulte "Base de datos relacional distribuida" en la página 43.

CREATE SERVER

Descripción

nombre-servidor

Nombra la fuente de datos que se está definiendo en la base de datos federada. El nombre no debe identificar una fuente de datos que esté descrita en el catálogo. El *nombre-servidor* no debe ser el mismo que el nombre de cualquier espacio de tablas de la base de datos federada.

TYPE *tipo-servidor*

Especifica el tipo de fuente de datos indicada por *nombre-servidor*. Esta opción es necesaria para los reiniciadores DRDA, SQLNET y NET8. Consulte “Apéndice F. Sistemas federados” en la página 1411 para obtener una lista de tipos de fuentes de datos soportados.

VERSION

Especifica la versión de la fuente de datos indicado por *nombre-servidor*.

versión

Especifica el número de versión. *versión* debe ser un entero.

release

Especifica el número de release de la versión indicada por *versión*. *release* debe ser un entero.

mod

Especifica el número de la modificación del release indicado por *release*. *mod* debe ser un entero.

constante-serie-versión

Especifica la designación completa de la versión. La *constante-serie-versión* puede ser un solo valor (por ejemplo, ‘8i’); o puede ser los valores concatenados de *versión*, *release* y, si se aplica, *mod* (por ejemplo, ‘8.0.3’).

WRAPPER *nombre-reiniciador*

Nombra el reiniciador que el servidor federado utiliza para interactuar con las fuentes de datos del tipo y versión indicados por *tipo-servidor* y ‘*versión-servidor*’.

AUTHORIZATION *nombre-autorización-remota*

Especifica el ID de autorización bajo el cual se realiza cualquier acción necesaria en la fuente de datos cuando se procesa la sentencia CREATE SERVER. Este ID debe tener la autorización (BINDADD o su equivalente) que las acciones necesarias necesitan.

PASSWORD *contraseña*

Especifica la contraseña asociada con el ID de autorización representado por *nombre-autorización-remota*. Si no se especifica *contraseña*, se usa por omisión la contraseña del ID que el usuario utilizó para conectarse a la base de datos federada.

OPTIONS

Indica las opciones de servidor que se han de habilitar. Consulte “Opciones de servidor” en la página 1415 para ver las descripciones de *nombre-opción-servidor* y sus valores.

ADD

Habilita una o varias opciones de servidor.

nombre-opción-servidor

Nombra una opción de servidor que se utilizará para configurar o proporcionar información acerca de la fuente de datos indicada por *nombre-servidor*.

constante-serie

Especifica un valor para *nombre-opción-servidor* como una constante de serie de caracteres.

Notas

- Si no se especifica *nombre-autorización-remota*, se utilizará el ID de autorización para la base de datos federada.
- La *contraseña* debe especificarse en letras mayúsculas/minúsculas que necesite la fuente de datos; si alguna letra de la *contraseña* debe estar en minúsculas, entrecomille la *contraseña*. Si se especifica un *identificador* pero no una *contraseña*, se supone que el tipo de autenticación de la fuente de datos indicada por *nombre-servidor* es CLIENT.
- Si se utiliza la sentencia CREATE SERVER para definir una instancia DB2 como fuente de datos, DB2 puede necesitar enlazar determinados paquetes con esa instancia. Si es necesario un enlace, el *nombre-autorización-remota* de la sentencia debe tener autorización para BIND. El tiempo necesario para realizar el enlace depende de la velocidad de la fuente de datos y de la velocidad de la conexión de red.
- Si una opción de servidor se establece en un valor para un tipo de fuente de datos y esta misma opción se establece en otro valor para una instancia de este tipo, el segundo valor prevalece sobre el primero para la instancia. Por ejemplo, suponga que PASSWORD se establece en ‘Y’ (sí, validar contraseñas en la fuente de datos) para las fuentes de datos DB2 Universal Database para OS/390 de un sistema federado. Más tarde, se utiliza el valor por omisión de esta opción (‘N’) para una fuente de datos DB2 Universal Database para OS/390 específica llamada SIBYL. Como resultado, las contraseñas se validarán en todas las fuentes de datos DB2 Universal Database para OS/390 excepto en SIBYL.

Ejemplos

Ejemplo 1: Defina una fuente de datos de DB2 para MVS/ESA 4.1 que sea accesible a través de un reiniciador llamado DB2WRAP. Llame a la fuente de datos CRANDALL. Además, especifique que:

CREATE SERVER

- MURROW y DROWSSAP serán el ID de autorización y la contraseña bajo los cuales se vinculan los paquetes en CRANDALL cuando se procesa esta sentencia.
- CRANDALL se define en la RDBMS de DB2 como una instancia llamada MYNODE.
- Cuando el servidor federado accede a CRANDALL, se conecta a una base de datos llamada MYDB.
- Los ID de autorización y las contraseñas bajo los que se puede acceder a CRANDALL se han de enviar a CRANDALL en mayúsculas.
- MYDB y la base de datos federada utilizan el mismo orden de clasificación.

```
CREATE SERVER CRANDALL
TYPE DB2/MVS
VERSION 4.1
WRAPPER DB2WRAP
AUTHORIZATION MURROW
PASSWORD DROWSSAP
OPTIONS
  ( NODE 'MYNODE',
    DBNAME 'MYDB',
    FOLD_ID 'U',
    FOLD_PW 'U',
    COLLATING_SEQUENCE 'Y' )
```

Ejemplo 2: Defina una fuente de datos Oracle 7.2 que se pueda acceder a través de un reiniciador llamado KLONDIKE. Llame a la fuente de datos CUSTOMERS. Especifique que:

- CUSTOMERS está definida en la RDBMS de Oracle como una instancia llamada ABC.

Proporcione estas estadísticas para el optimizador:

- La CPU para el servidor federado se ejecuta el doble de rápido que la CPU que soporta CUSTOMERS.
- Los dispositivos de E/S del servidor federado procesan los datos 1,5 veces más rápido que los dispositivos de E/S de CUSTOMERS.

```
CREATE SERVER CUSTOMERS
TYPE ORACLE
VERSION 7.2
WRAPPER KLONDIKE
OPTIONS
  ( NODE 'ABC',
    CPU_RATIO '2.0',
    IO_RATIO '1.5' )
```

CREATE TABLE

La sentencia CREATE TABLE define una tabla. Esta definición debe incluir el nombre de la tabla y los nombres y atributos de sus columnas. Puede incluir otros atributos de la tabla, como puede ser la clave primaria o restricciones de comprobación.

Invocación

Esta sentencia se puede intercalar en un programa de aplicación o se puede emitir utilizando sentencias de SQL dinámicas. Es una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

Autorización

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Autorización CREATETAB para la base de datos y privilegio USE para el espacio de tablas, y también uno de los elementos siguientes:
 - Autorización IMPLICIT_SCHEMA para la base de datos, si no existe el nombre de esquema implícito ni explícito de la tabla
 - Privilegio CREATEIN para el esquema, si el nombre de esquema de la tabla hace referencia a un esquema existente.

Si se define una subtabla, el ID de autorización debe ser el mismo que el definidor de la tabla raíz de la jerarquía de tablas.

Para definir una clave foránea, los privilegios del ID de autorización de la sentencia deben incluir uno de los siguientes en la tabla padre:

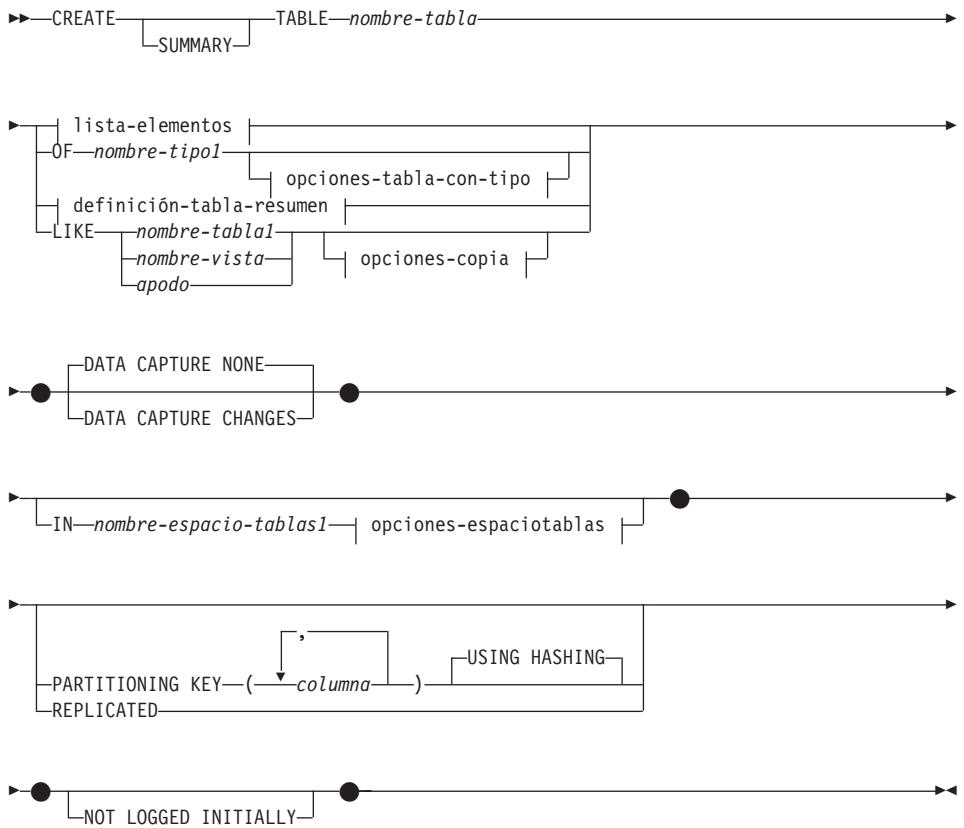
- El privilegio REFERENCES en la tabla
- El privilegio REFERENCES en todas las columnas de la clave padre especificada
- Privilegio CONTROL para la tabla
- Autorización SYSADM o DBADM.

Para definir una tabla de resumen (utilizando una selección completa), el ID de autorización de la sentencia debe tener al menos uno de los siguientes privilegios en cada tabla o vista identificada en la selección completa:

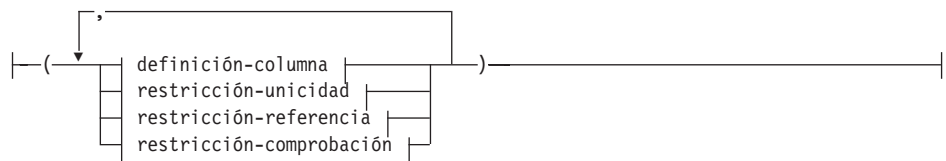
- Privilegio SELECT para la tabla o vista y el privilegio ALTER si se especifica REFRESH DEFERRED o REFRESH IMMEDIATE
- Privilegio CONTROL para la tabla o vista
- Autorización SYSADM o DBADM.

CREATE TABLE

Sintaxis



lista-elementos:



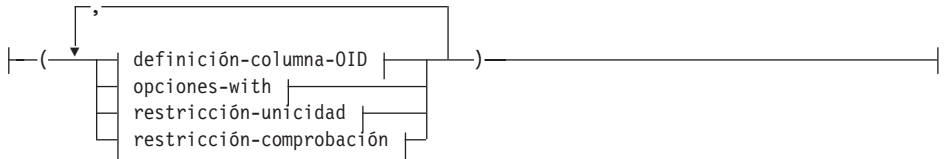
opciones-tabla-con-tipo:



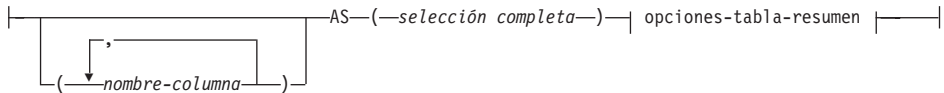
cláusula-under:

|—UNDER—*nombre-supertabla*—INHERIT SELECT PRIVILEGES—|

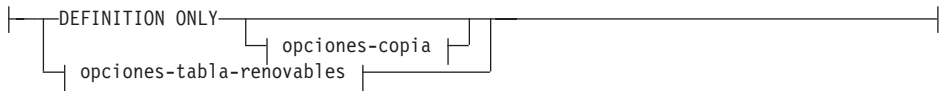
lista-elementos-tipo:



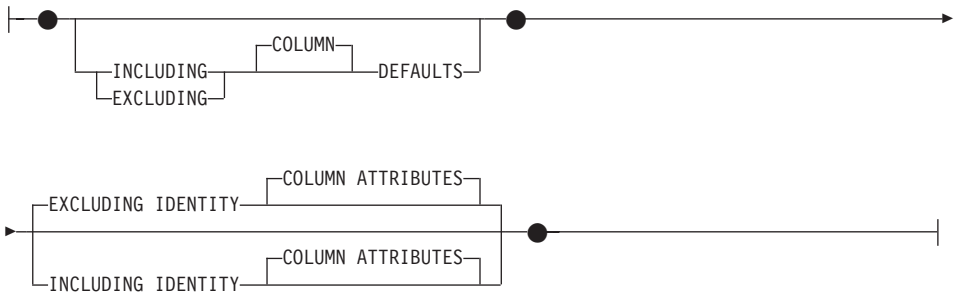
definición-tabla-resumen:



opciones-tabla-resumen:



opciones-copia:

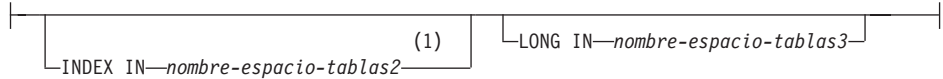


CREATE TABLE

opciones-tabla-renovables:



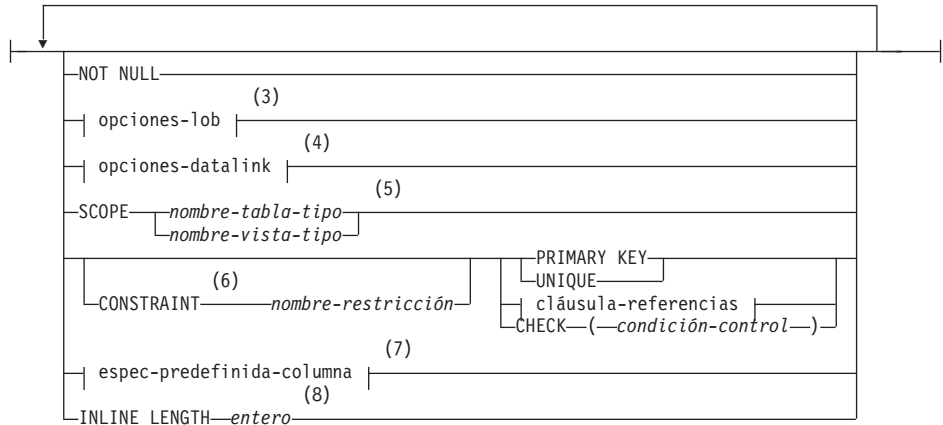
opciones-espaciotablas:



definición-columna:



opciones-columna:



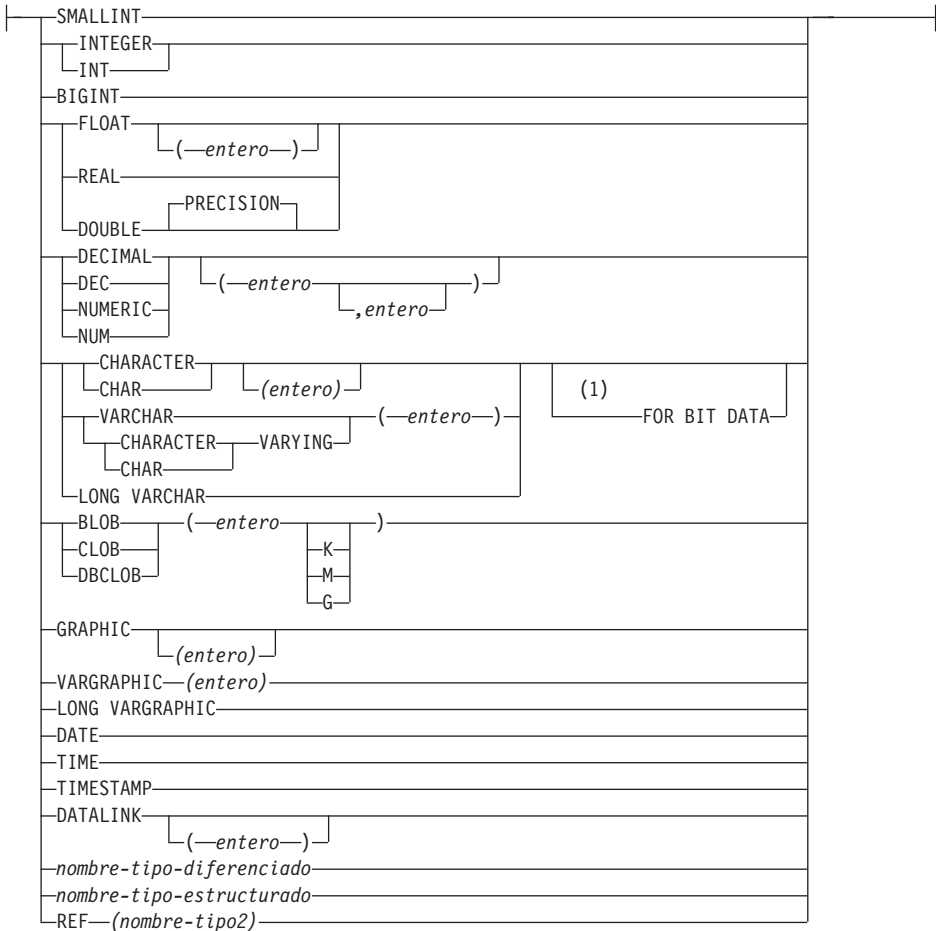
Notas:

- 1 Sólo se puede especificar el espacio de tablas que contendrá un índice de tabla cuando se crea la tabla.
- 2 Si la primera opción de columna elegida es una especificación de columna generada, se puede omitir el tipo de datos. Éste se determinará a partir del tipo de datos resultante de la expresión de generación.

- 3 La cláusula opciones-lob sólo se aplica a tipos de gran objeto (BLOB, CLOB y DBCLOB) y a los tipos diferenciados basados en tipos de gran objeto.
- 4 La cláusula opciones-datalink sólo se aplica al tipo DATALINK y a los tipos diferenciados basados en el tipo DATALINK. Se necesita la cláusula LINKTYPE URL para estos tipos.
- 5 La cláusula SCOPE sólo se aplica al tipo REF.
- 6 Para que haya compatibilidad con la Versión 1, se puede omitir la palabra clave CONSTRAINT en una *definición-columna* que defina una cláusula-referencias.
- 7 Los atributos de columna IDENTITY no pueden utilizarse en una base de datos EEE (Extended Enterprise Edition) que tenga más de una partición.
- 8 INLINE LENGTH sólo es aplicable a columnas definidas como tipos estructurados.

tipo-datos:

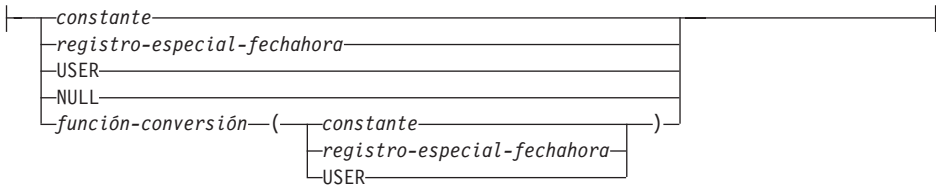
CREATE TABLE



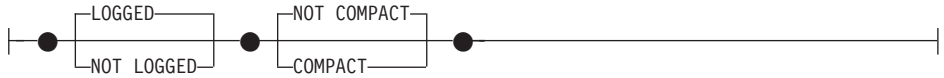
Notas:

- 1 La cláusula FOR BIT DATA puede especificarse en orden aleatorio con las restricciones de columna siguientes.

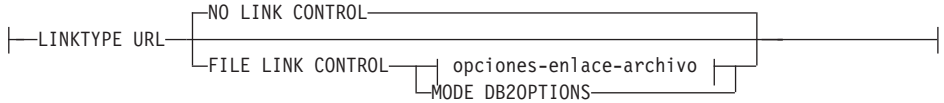
valores-omisión:



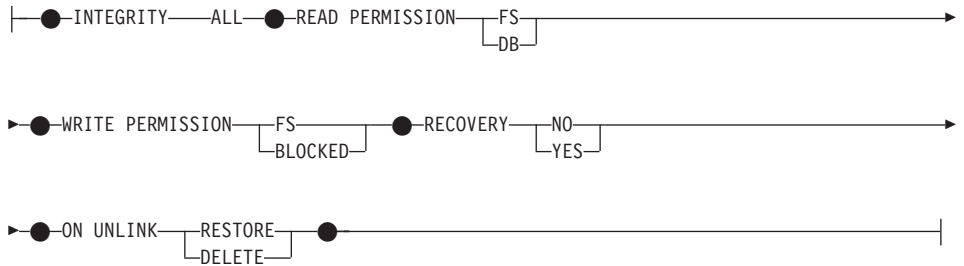
opciones-lob:



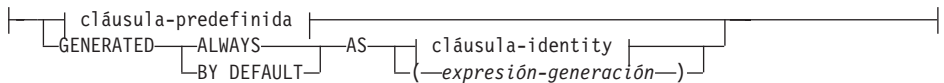
opciones-datalink:



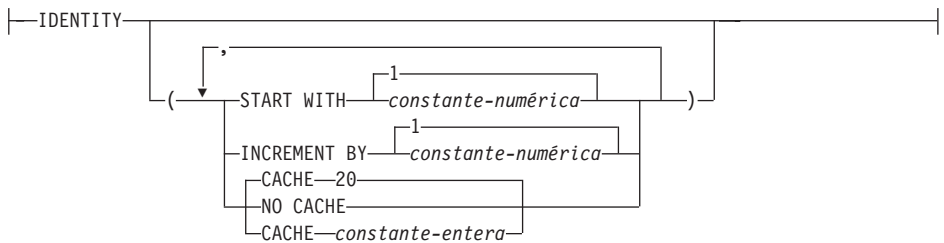
opciones-enlace-archivo:



espec-predefinida-columna:

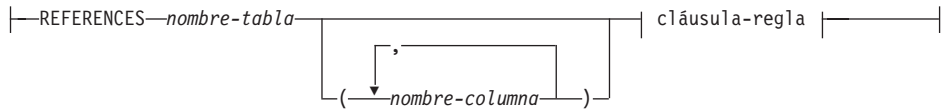


cláusula-identity:

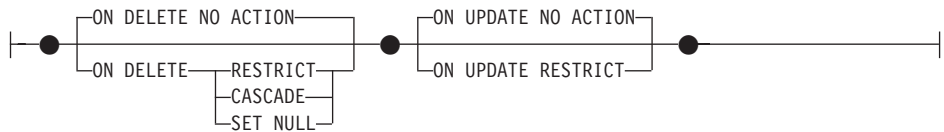


CREATE TABLE

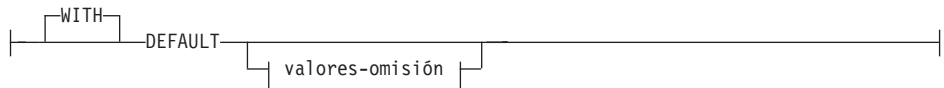
cláusula-referencias:



cláusula-regla:



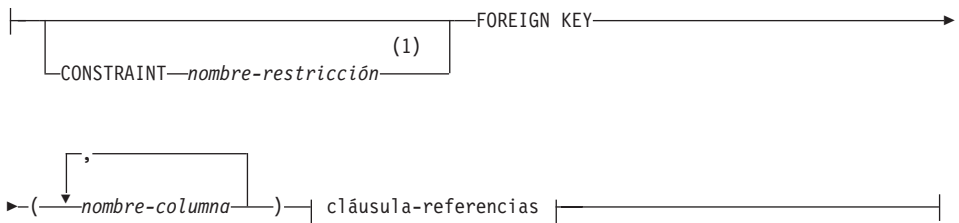
cláusula-predefinida:



restricción-unicidad:



restricción-referencia:



restricción-comprobación:



definición-columna-OID:

```
|—REF IS—nombre-columna-OID—USER GENERATED—|
```

opciones-with:

```
|—nombre-columna—WITH OPTIONS—| opciones-columna |
```

Notas:

- 1 Por razones de compatibilidad con la Versión 1, el nombre-restricción puede especificarse seguido de FOREIGN KEY (sin la palabra clave CONSTRAINT).

Descripción**SUMMARY**

Indica que se va a definir una tabla de resumen. La palabra clave es opcional, pero si se especifica, la sentencia debe incluir una *definición-tabla-resumen* (SQLSTATE 42601).

nombre-tabla

Indica el nombre de la tabla. El nombre, incluido el calificador implícito o explícito, no debe designar una tabla, vista o seudónimo descritos en el catálogo. El nombre de esquema no debe ser SYSIBM, SYSCAT, SYSFUN ni SYSSTAT (SQLSTATE 42939).

OF *nombre-tipo1*

Especifica que las columnas de la tabla están basadas en los atributos del tipo estructurado identificado por *nombre-tipo1*. Si se especifica *nombre-tipo1* sin un nombre de esquema, el nombre de tipo se resuelve efectuando una búsqueda en los esquemas de la vía de acceso de SQL (definida por la opción de preproceso FUNCPATH en el caso del SQL estático y por el registro CURRENT PATH en el caso de SQL dinámicas). El nombre de tipo debe ser el nombre de un tipo existente definido por el usuario (SQLSTATE 42704) y debe ser un tipo estructurado del que se puede crear una instancia (SQLSTATE 428DP), con al menos un atributo (SQLSTATE 42997).

Si no se especifica UNDER, debe especificarse una columna de identificador de objeto (consulte la *definición-columna-OID*). Esta columna de identificador de objeto es la primera columna de la tabla. La columna del ID de objeto va seguida de columnas basadas en los atributos de *nombre-tipo1*.

HIERARCHY *nombre-jerarquía*

Indica la tabla de jerarquía asociada con la jerarquía de tabla. Se crea a la vez que la tabla raíz de la jerarquía. Los datos para todas las subtablas en

CREATE TABLE

la jerarquía de tablas con tipo se almacenan en la tabla de jerarquía. No se puede hacer referencia a una tabla de jerarquía directamente en una sentencia de SQL. Un *nombre-jerarquía* es un *nombre-tabla*. El *nombre-jerarquía*, incluido el nombre de esquema implícito o explícito, no debe identificar una tabla, apodo, vista o seudónimo descritos en el catálogo. Si se especifica el nombre de esquema, debe ser el mismo que el nombre de esquema de la tabla que se está creando (SQLSTATE 428DQ). Si se omite esta cláusula al definir la tabla raíz, el sistema genera un nombre que consiste en el nombre de la tabla que se está creando seguido de un sufijo exclusivo de modo que el identificador es exclusivo dentro de los identificadores de las tablas, vistas, seudónimo y apodos existentes.

UNDER *nombre-supertabla*

Indica que la tabla es una subtabla de *nombre-supertabla*. La supertabla debe ser una tabla existente (SQLSTATE 42704) y la tabla debe estar definida mediante un tipo estructurado que sea el supertipo inmediato de *nombre-tipo1* (SQLSTATE 428DB). El nombre de esquema de *nombre-tabla* y *nombre-supertabla* debe ser el mismo (SQLSTATE 428DQ). La tabla identificada por *nombre-supertabla* no debe tener ninguna subtabla existente ya definida mediante *nombre-tipo1* (SQLSTATE 42742).

Entre las columnas de la tabla, se incluye la columna de identificador de objeto de la supertabla con su tipo modificado para que sea *REF(nombre-tipo1)*, seguida de columnas basadas en los atributos de *nombre-tipo1* (recuerde que el tipo incluye los atributos de su supertipo). Los atributos no pueden tener el mismo nombre que la columna de OID (SQLSTATE 42711).

No pueden especificarse otras opciones de tabla como el espacio de tablas, DATA CAPTURE, NOT LOGGED INITIALLY y la clave de particionamiento. Estas opciones se heredan de la supertabla (SQLSTATE 42613).

INHERIT SELECT PRIVILEGES

Cualquier usuario o grupo que sostenga un privilegio SELECT sobre la supertabla recibirá un privilegio equivalente sobre la subtabla recién creada. Se considera que el definidor de la subtabla es el encargado de otorgar este privilegio.

lista-elementos

Define los elementos de una tabla. Esto incluye la definición de las columnas y las restricciones de la tabla.

lista-elementos-tipo

Define los elementos adicionales de una tabla con tipo. Esto incluye las opciones adicionales para las columnas, la adición de una columna de identificador de objeto (sólo la tabla raíz) y las restricciones de la tabla.

definición-tabla-resumen

Si la definición de tabla se basa en el resultado de una consulta, la tabla es una tabla de resumen basada en la consulta.

nombre-columna

Designa las columnas de la tabla. Si se especifica una lista de nombres de columna, ésta debe constar de tantos nombres como columnas haya en la tabla resultante de la selección completa. Cada *nombre-columna* debe ser exclusivo y no calificado. Si no se especifica una lista de nombres de columnas, las columnas de la tabla heredan los nombres de las columnas de la tabla resultante de la selección completa.

Debe especificarse una lista de nombres de columna si la tabla resultante de la selección completa tiene nombres de columna duplicados o una columna sin nombre (SQLSTATE 42908). Una columna sin nombre es una columna derivada de una constante, función, expresión u operación de conjuntos que no se designa utilizando la cláusula AS de la lista de selección.

AS

Introduce la consulta que se utiliza para la definición de la tabla y para determinar los datos incluidos en la tabla.

selección completa

Define la consulta en la que se basa la tabla. Las definiciones de columna resultantes son las mismas que las de una vista definida con la misma consulta.

Cada elemento de la lista de selección debe tener un nombre (utilice la cláusula AS para las expresiones; vea la página “cláusula-select” en la página 473 para obtener detalles). Las *opciones-tabla-resumen* especificadas definen los atributos de la tabla de resumen. La opción elegida también define el contenido de la selección completa de la manera siguiente.

Cuando se especifica DEFINITION ONLY, puede especificarse cualquier selección completa válida que no haga referencia a una tabla con tipo o a una vista con tipo.

Cuando se especifica REFRESH DEFERRED o REFRESH IMMEDIATE, la selección completa no puede incluir (SQLSTATE 428EC):

- referencias a un apodo, una tabla de resumen, una tabla temporal declarada o una tabla con tipo en ninguna cláusula FROM
- referencias a una vista donde la selección completa de la vista viola cualquiera de las restricciones listadas en la selección completa de la tabla de resumen
- expresiones que sean un tipo de referencia o un tipo DATALINK (o tipos diferenciados basados en estos tipos)

CREATE TABLE

- funciones que tengan acción externa
- funciones escritas en SQL
- funciones que dependan de características físicas (por ejemplo, NODENUMBER, PARTITION)
- referencias de tablas o vistas a objetos del sistema (tampoco deben especificarse tablas explain)
- expresiones que son un tipo estructurado o un tipo LOB (o un tipo diferenciado basado en un tipo LOB)

Cuando se especifica REFRESH IMMEDIATE:

- la selección completa debe ser una subselección
- la subselección no puede incluir:
 - funciones que no sean deterministas
 - selecciones completas escalares
 - predicados con selecciones completas
 - registros especiales
- debe incluirse una cláusula GROUP BY en la subselección a menos que la tabla de resumen sea REPLICATED.
- Las funciones de columna soportadas son SUM, COUNT, COUNT_BIG y GROUPING (sin DISTINCT). La lista seleccionada debe contener una columna COUNT(*) o COUNT_BIG(*). Si la lista de selección de tabla contiene SUM(X) donde X es un argumento con posibilidad de tener nulos, entonces la tabla de resumen también debe tener COUNT(X) en su lista de selección. Estas funciones de columna no pueden formar parte de ninguna expresión.
- Si la cláusula FROM hace referencia a más de una tabla o vista, sólo puede definir una unión interna sin utilizar la sintaxis INNER JOIN explícita
- todos los elementos GROUP BY deben incluirse en la lista de selección
- No pueden utilizarse GROUPING SETS, CUBE ni ROLLUP. Los elementos de GROUP BY y las funciones de columna GROUPING asociadas, contenidos en la lista de selección, deben formar una clave de unicidad del conjunto resultante. Por tanto, son aplicables las restricciones siguientes:
 - no puede haber conjuntos de agrupación repetidos. Por ejemplo, ROLLUP(X,Y), X no está permitido, pues es equivalente a GROUPING SETS((X,Y),(X),(X))
 - si X es un elemento de GROUP BY que puede ser nulo y aparece dentro de GROUPING SETS, CUBE o ROLLUP, entonces GROUPING(X) debe aparecer en la lista de selección

- no está permitido hacer agrupaciones de acuerdo con constantes
- no está permitido utilizar una cláusula HAVING
- si se utiliza un grupo de nodos de varias particiones, la clave de particionamiento debe ser un subconjunto de los elementos de GROUP BY, o la tabla de resumen debe estar replicada.

opciones-tabla-resumen

Definen los atributos de la tabla de resumen.

DEFINITION ONLY

La consulta sólo se utiliza para definir la tabla. La tabla no se rellena con los resultados de la consulta y no puede utilizarse la sentencia REFRESH TABLE. Cuando se completa la sentencia CREATE TABLE, la tabla ya no se considera una tabla de resumen.

Se definen las columnas de la tabla basándose en las definiciones de las columnas que resultan de la selección completa. Si la selección completa hace referencia a una tabla individual de la cláusula FROM, los elementos de la lista de selección que son columnas de esa tabla se definen utilizando el nombre de columna, tipo de datos y posibilidad de contener nulos de la tabla referenciada.

opciones-tabla-renovables

Defina las opciones renovables de los atributos de la tabla de resumen.

DATA INITIALLY DEFERRED

Los datos no se insertan en la tabla como parte de la sentencia CREATE TABLE. Se utiliza una sentencia REFRESH TABLE que especifique el *nombre-tabla* para insertar los datos en la tabla.

REFRESH

Indica cómo se mantienen los datos de la tabla.

DEFERRED

Los datos de la tabla pueden renovarse en cualquier momento mediante la sentencia REFRESH TABLE. Los datos de la tabla sólo reflejan el resultado de la consulta como una instantánea en el momento en que se procesa la sentencia REFRESH TABLE. Las tablas de resumen definidas con este atributo no permiten las sentencias INSERT, UPDATE o DELETE (SQLSTATE 42807).

IMMEDIATE

Los cambios realizados en las tablas principales como parte de una sentencia DELETE, INSERT o UPDATE se aplican en cascada a la tabla de resumen. En este caso, el contenido de la tabla, en cualquier momento, es igual que cuando se procesa

CREATE TABLE

la *subselección* especificada. Las tablas de resumen definidas con este atributo no permiten las sentencias INSERT, UPDATE o DELETE (SQLSTATE 42807).

ENABLE QUERY OPTIMIZATION

La tabla de resumen se puede utilizar para la optimización de consultas bajo las condiciones apropiadas.

DISABLE QUERY OPTIMIZATION

La tabla de resumen no se utilizará para la optimización de consultas. La tabla todavía puede consultarse directamente.

LIKE *nombre-tabla1* **o** *nombre-vista* **o** *apodo*

Especifica que las columnas de la tabla tienen exactamente el mismo nombre y descripción que las columnas de la tabla identificada (*nombre-tabla1*), vista (*nombre-vista*) o apodo (*apodo*). El nombre especificado después de LIKE debe identificar una tabla, vista o apodo existentes en el catálogo, o una tabla temporal declarada. No se puede especificar una tabla con tipo ni una vista con tipo (SQLSTATE 428EC).

El uso de LIKE es una definición implícita de *n* columnas, donde *n* es el número de columnas de la tabla, vista o apodo designados.

- Si se designa una tabla, la definición implícita incluye el nombre de columna, tipo de datos y posibilidad de contener nulos de cada columna de *nombre-tabla1*. Si no se especifica EXCLUDING COLUMN DEFAULTS, también se incluye el valor por omisión de la columna.
- Si se designa una vista, la definición implícita incluye el nombre de columna, tipo de datos y posibilidad de contener nulos de cada columna resultante de la selección completa definida en *nombre-vista*.
- Si se designa una apodo, la definición implícita incluye el nombre de columna, tipo de datos y posibilidad de contener nulos de cada columna de *apodo*.

Según cuáles sean las cláusulas de atributos de copia, se pueden incluir o excluir el valor por omisión de la columna y los atributos IDENTITY de la columna. La definición implícita no incluye ningún otro atributo de la tabla, vista o apodo designados. Por lo tanto, la nueva tabla no tiene ninguna restricción de unicidad, restricción de clave foránea, desencadenantes ni índices. La tabla se crea en el espacio de tablas implícita o explícitamente especificado mediante la cláusula IN y la tabla tiene cualquier otra cláusula opcional sólo si la cláusula opcional se especifica.

opciones-copia

Estas opciones especifican si deben copiarse atributos adicionales de la definición de la tabla resultante origen (tabla, vista o selección completa).

INCLUDING COLUMN DEFAULTS

Especifica que deben copiarse los valores por omisión de cada columna actualizable contenida en la definición de la tabla resultante origen. Las columnas que no son actualizables no tienen un valor por omisión definido en la correspondiente columna de la tabla creada.

Si se especifica LIKE *nombre-tabla* y *nombre-tabla* designa una tabla base o una tabla temporal declarada, la opción por omisión es INCLUDING COLUMN DEFAULTS.

EXCLUDING COLUMN DEFAULTS

Especifica que no deben copiarse los valores por omisión de las columnas de la tabla resultante origen.

Esta es la cláusula por omisión, excepto si se especifica LIKE *nombre-tabla* y *nombre-tabla* designa una tabla base o una tabla temporal declarada.

INCLUDING IDENTITY COLUMN ATTRIBUTES

Especifica que deben copiarse, si es posible, los atributos de columna de identidad (valores START WITH, INCREMENT BY y CACHE) contenidos en la definición de la tabla resultante origen. Es posible copiar los atributos de columna de identidad, si el elemento de la correspondiente columna de la tabla, vista o selección completa es el nombre de una columna de tabla o de vista que, directa o indirectamente, se correlaciona con el nombre de una columna de tabla base que tiene el atributo de identidad. En todos los demás casos, las columnas de la nueva tabla no obtendrán el atributo de identidad. Por ejemplo:

- La lista de selección de la selección completa contiene varias instancias de un nombre de columna de identidad (es decir, se selecciona la misma columna más de una vez)
- la lista de selección de la selección completa contiene varias columnas de identidad (es decir, supone la ejecución de una operación de unión)
- la columna de identidad está contenida en una expresión de la lista de selección
- la selección completa contiene una operación de conjuntos (union, except o intersect).

EXCLUDING IDENTITY COLUMN ATTRIBUTES

Especifica que no deben copiarse los atributos de columna de identidad contenidos en la definición de la tabla resultante origen.

definición-columna

Define los atributos de una columna.

CREATE TABLE

nombre-columna

Es el nombre de una columna de la tabla. El nombre no puede estar calificado y no se puede utilizar el mismo nombre para más de una columna de la tabla.

Una tabla puede tener las características siguientes:

- un tamaño de página de 4K con un máximo de 500 columnas en que la suma de las cuentas de bytes de las columnas no debe ser superior a 4005 en un tamaño de página de 4K. Consulte el “Tamaño de fila” en la página 885 para obtener más detalles.
- un tamaño de página de 8K con un máximo de 1012 columnas en que la suma de las cuentas de bytes de las columnas no debe ser superior a 8101. Consulte el “Tamaño de fila” en la página 885 para obtener más detalles.
- un tamaño de página de 16K con un máximo de 1.012 columnas en que la suma de las cuentas de bytes de las columnas no debe ser superior a 16.293.
- un tamaño de página de 32K con un máximo de 1.012 columnas en que la suma de las cuentas de bytes de las columnas no debe ser superior a 32.677.

tipo-datos

Se trata de uno de los tipos de la lista siguiente. Utilice:

SMALLINT

Para especificar un entero pequeño.

INTEGER o INT

Para especificar un entero grande.

BIGINT

Para especificar un entero superior.

FLOAT(*entero*)

Para especificar un número de coma flotante de precisión simple o doble, en función del valor del *entero*. El valor del entero debe estar en el rango 1-53. Los valores 1 a 24 indican precisión simple y los valores 25 a 53 indican precisión doble.

También puede especificar:

REAL

Para especificar un valor de coma flotante de precisión simple.

DOUBLE

Para especificar coma flotante de precisión doble.

DOUBLE-PRECISION

Para especificar coma flotante de precisión doble.

FLOAT

Para especificar coma flotante de precisión doble.

DECIMAL(*entero-precisión, entero-escala*) o **DEC**(*entero-precisión, entero-escala*)

Para especificar un número decimal. El primer entero es la precisión del número (es decir, el número total de dígitos); su valor varía entre 1 y 31. El segundo entero es la escala del número (es decir, el número de dígitos situados a la derecha de la coma decimal); su valor varía entre 0 y la precisión del número.

Si no se especifican la precisión ni la escala, se emplean los valores por omisión 5,0. Las palabras **NUMERIC** y **NUM** pueden utilizarse como sinónimos de **DECIMAL** y **DEC**.

CHARACTER(*entero*) o **CHAR**(*entero*) o **CHARACTER** o **CHAR**

Para una serie de caracteres de longitud fija de longitud *entero*, que puede oscilar entre 1 y 254. Si se omite la especificación de la longitud, se supone que la longitud es de 1 carácter.

VARCHAR(*entero*), o **CHARACTER VARYING**(*entero*), o **CHAR VARYING**(*entero*)

Para una serie de caracteres de longitud variable de *entero* de longitud máxima, que puede estar en el rango de 1 a 32 672.

LONG VARCHAR

Para una serie de caracteres de longitud variable con una longitud máxima de 32700.

FOR BIT DATA

Especifica que el contenido de la columna se tratará como datos de bit (binarios). Durante el intercambio de datos con otros sistemas, no se efectúan conversiones de página de códigos. Las comparaciones se efectúan en binario, sin tener en cuenta el orden de clasificación de la base de datos.

BLOB(*entero [K | M | G]*)

Para una serie de gran objeto binario de la longitud máxima especificada en bytes.

La longitud puede oscilar entre 1 y 2 147 483 647 bytes.

Si el *entero* ya está especificado por sí solo, se entiende que esa es la longitud máxima.

Si se especifica *entero K* (ya sea en mayúsculas o en minúsculas), la longitud máxima es el *entero* multiplicado por 1 024. El valor máximo del *entero* es 2 097 152.⁷⁵

Si se especifica *entero M*, la longitud máxima es el *entero* por 1 048 576. El valor máximo de *entero* es 2 048.⁷⁵

75. Cuando se proporciona un múltiplo de K, M o G que da como cálculo 2 147 483 648, el valor real que se utiliza es 2 147 483 647 (o 2 gigabytes menos 1 byte), que es la longitud máxima para una columna LOB.

CREATE TABLE

Si se especifica *entero* G, la longitud máxima es el *entero* multiplicado por 1 073 741 824. El valor máximo de *entero* es 2.⁷⁵

Para crear series BLOB con más de 1 gigabyte, debe especificar la opción NOT LOGGED.

Entre el *entero* y K, M o G se permite poner cualquier número de espacios. Tampoco se necesita ningún espacio. Serían válidos, por ejemplo, los siguientes valores.

BLOB(50 K) BLOB(50 K) BLOB (50 K)

CLOB(*entero* [K | M | G])⁷⁶

Para una serie de gran objeto de caracteres de la longitud máxima especificada en bytes.

El significado de *entero* K | M | G es el mismo que para BLOB.

Para crear series CLOB con más de 1 gigabyte, debe especificar la opción NOT LOGGED.

DBCLOB(*entero* [K | M | G])

Para una serie de gran objeto de caracteres de doble byte de la longitud máxima especificada en caracteres de doble byte.

El significado de *entero* K | M | G es parecido al de BLOB. La diferencia radica en que el número especificado es el número de caracteres de doble byte y el tamaño máximo es de 1 073 741 823 caracteres de doble byte.

Para crear series DBCLOB con más de 1 gigabyte, debe especificar la opción NOT LOGGED.

GRAPHIC(*entero*)

Para una serie gráfica de longitud fija de longitud *entero*, que puede ir de 1 a 127. Si se omite la especificación de longitud, se supone una longitud de 1.

VARGRAPHIC(*entero*)

Para una serie gráfica de longitud variable de *entero* de longitud máxima, que puede estar en el rango de 1 a 16 336.

LONG VARGRAPHIC

Para una serie gráfica de longitud variable con una longitud máxima de 16 350.

DATE

Para la fecha.

76. Tenga en cuenta que no es posible especificar la cláusula FOR BIT DATA para las columnas CLOB. No obstante, se puede asignar una serie CHAR FOR BIT DATA a una columna CLOB y se puede concatenar una serie CHAR FOR BIT DATA con la serie CLOB.

TIME

Para la hora.

TIMESTAMP

Para la indicación horaria.

DATALINK o **DATALINK**(*entero*)

Para un enlace con datos almacenados fuera de la base de datos.

La columna de la tabla consta de "valores ancla" que contienen la información de referencia que se necesita para establecer y mantener el enlace con los datos externos así como un comentario opcional.

La longitud de una columna DATALINK es de 200 bytes. Si se especifica *entero*, debe ser 200. Si se omite la especificación de longitud, se supone una longitud de 200 bytes.

Un valor DATALINK es un valor encapsulado con un conjunto de funciones escalares incorporadas. Hay una función denominada DLVALUE que sirve para crear un valor DATALINK. Pueden utilizarse las funciones siguientes para extraer atributos de un valor DATALINK.

- DLCOMMENT
- DLLINKTYPE
- DLURLCOMPLETE
- DLURLPATH
- DLURLPATHONLY
- DLURLSCHEME
- DLURLSERVER

Una columna DATALINK tiene las restricciones siguientes:

- La columna no puede formar parte de ningún índice. Por lo tanto, no puede incluirse como columna de una clave primaria ni de una restricción de unicidad (SQLSTATE 42962).
- La columna no puede ser una clave foránea de una restricción de referencia (SQLSTATE 42830).
- No puede especificarse un valor por omisión (WITH DEFAULT) para la columna. Si la columna puede ser nula, el valor por omisión para la columna es NULL (SQLSTATE 42894).

nombre-tipo-diferenciado

Para un tipo definido por el usuario que es un tipo diferenciado. Si se especifica un nombre de tipo diferenciado sin un nombre de esquema, el nombre del tipo diferenciado se resuelve efectuando una búsqueda en los esquemas de la vía de acceso de SQL

CREATE TABLE

(definida por la opción de preproceso FUNCPATH en el caso del SQL estático y por el registro CURRENT PATH en el caso de SQL dinámicas).

Si se define una columna con un tipo diferenciado, el tipo de datos de la columna es el tipo diferenciado. La longitud y la escala de la columna son, respectivamente, la longitud y la escala del tipo fuente del tipo diferenciado.

Si una columna definida con un tipo diferenciado es la clave externa de una restricción de referencia, el tipo de datos de la columna correspondiente de la clave primaria debe tener el mismo tipo diferenciado.

nombre-tipo-estructurado

Para especificar un tipo definido por el usuario que es un tipo estructurado. Si se especifica un nombre de tipo estructurado sin un nombre de esquema, el nombre de tipo estructurado se resuelve buscando en los esquemas especificados en la vía de acceso de SQL (definida por la opción de preproceso FUNCPATH en el SQL estático y por el registro CURRENT PATH en el SQL dinámico).

Si se define una columna utilizando un tipo estructurado, el tipo de datos estáticos de la columna es el tipo estructurado. La columna puede contener valores con un tipo dinámico que es un subtipo de *nombre-tipo-estructurado*.

Si se define una columna utilizando un tipo estructurado, la columna no se puede utilizar en una clave primaria, restricción de unicidad, clave foránea, clave de índice ni clave de particionamiento (SQLSTATE 42962).

Si se define una columna utilizando un tipo estructurado, y la columna contiene un atributo de tipo de referencia, a cualquier nivel de anidamiento, ese atributo no tiene ámbito. Para utilizar un atributo de esa clase en una operación de desreferencia, es necesario especificar explícitamente un ámbito (SCOPE), utilizando una especificación CAST.

Si se define una columna utilizando un tipo estructurado con un atributo de tipo DATALINK, o un tipo diferenciado derivado de DATALINK, esta columna sólo puede ser nula. Si se intenta utilizar la función constructora para este tipo, se obtendrá un error (SQLSTATE 428ED), y por tanto no se puede insertar ninguna instancia de este tipo en la columna.

REF (*nombre-tipo2*)

Para una referencia a una tabla con tipo. Si se especifica *nombre-tipo2* sin un nombre de esquema, el nombre de tipo se

resuelve efectuando una búsqueda en los esquemas de la vía de acceso de SQL (definida por la opción de preproceso FUNCSPATH en el caso de SQL estático y por el registro CURRENT PATH en el caso del SQL dinámico). El tipo de datos subyacente de la columna está basado en el tipo de datos de representación especificado en la cláusula REF USING de la sentencia CREATE TYPE para *nombre-tipo2* o el tipo raíz de la jerarquía de tipos de datos donde está incluido *nombre-tipo2*.

opciones-columna

Define opciones adicionales relacionadas con las columnas de la tabla.

NOT NULL

Evita que la columna contenga valores nulos.

Si no se especifica NOT NULL, la columna puede contener valores nulos, y su valor por omisión es un valor nulo o bien el valor proporcionado por la cláusula WITH DEFAULT.

opciones-lob

Especifica opciones para los tipos de datos LOB.

LOGGED

Especifica que los cambios efectuados en la columna han de anotarse en la anotación cronológica. Acto seguido, los programas de utilidad de la base de datos (como RESTORE DATABASE) pueden recuperar los datos de estas columnas. LOGGED es el valor por omisión.

Los LOB superiores a 1 gigabyte no pueden registrarse (SQLSTATE 42993) y los LOB superiores a 10 megabytes no deberían registrarse.

NOT LOGGED

Especifica que los cambios efectuados en la columna no se van a anotar cronológicamente.

NOT LOGGED no tiene ningún efecto en una operación de confirmación o retroacción; es decir, la coherencia de la base de datos se mantiene incluso si una transacción se retrotrae, no importa si se anota cronológicamente el valor LOB o no. La implicación de no efectuar el registro es que durante una operación de avance, tras una operación de carga o de copia de seguridad, los datos de LOB se sustituirán por ceros en todos aquellos valores de LOB que hubieran generado entradas de la anotación cronológica que se habrían reproducido durante el avance. Durante la recuperación de una colisión, todos los cambios confirmados y los cambios retrotraídos reflejarán los resultados esperados. Consulte el

CREATE TABLE

manual *Administration Guide* para las implicaciones de no anotar cronológicamente las columnas LOB.

COMPACT

Especifica que los valores de la columna LOB deben ocupar el mínimo espacio de disco (liberar las páginas de disco sobrantes del último grupo utilizado por el valor LOB), en lugar de dejar el espacio restante al final del espacio de almacenamiento de LOB que podría facilitar operaciones posteriores de adición. Observe que almacenar datos de esta manera puede influir negativamente en el rendimiento de cualquier operación de adición (aumento de longitud) que se lleve a cabo en la columna.

NOT COMPACT

Especifica cierto espacio para las inserciones a fin de facilitar los cambios futuros que se efectúen en los valores LOB de la columna. Éste es el valor por omisión.

opciones-datalink

Especifica las opciones asociadas con un tipo de datos DATALINK.

LINKTYPE URL

Define el tipo de enlace como un Localizador de recursos uniformes (URL).

NO LINK CONTROL

Especifica que no se realizará ninguna comprobación para determinar si existe el archivo. Sólo se comprobará la sintaxis del URL. El gestor de bases de datos no tiene control sobre el archivo.

FILE LINK CONTROL

Especifica que debe comprobarse la existencia del archivo. Se pueden utilizar opciones adicionales para proporcionar al gestor de bases de datos un control mayor sobre el archivo.

opciones-enlace-archivo

Opciones adicionales para definir el nivel de control de gestor de bases de datos del enlace del archivo.

INTEGRITY

Especifica el nivel de integridad del enlace entre un valor DATALINK y el archivo real.

ALL

Cualquier archivo especificado como un valor DATALINK está bajo el control del gestor de bases de

datos y NO puede suprimirse ni red denominarse mediante interfaces de programación de sistema de archivos de tipo estándar.

READ PERMISSION

Especifica cómo se determina el permiso para leer el archivo especificado en un valor DATALINK.

FS El permiso de acceso de lectura está determinado por los permisos del sistema de archivos. Se puede acceder a estos archivos sin recuperar el nombre de archivo de la columna.

DB

El permiso de acceso de lectura está determinado por la base de datos. Sólo se permitirá el acceso al archivo si se pasa un símbolo de accesos de archivo válido, devuelto en la recuperación del valor DATALINK de la tabla, en la operación de la apertura.

WRITE PERMISSION

Especifica cómo se determina el permiso para grabar en el archivo especificado en un valor DATALINK.

FS El permiso de acceso de grabación está determinado por los permisos del sistema de archivos. Se puede acceder a estos archivos sin recuperar el nombre de archivo de la columna.

BLOCKED

El acceso de grabación está bloqueado. El archivo no puede actualizarse directamente mediante ninguna interfaz. Debe utilizarse un mecanismo alternativo para que puedan realizarse actualizaciones en la información. Por ejemplo, se copia el archivo, se actualiza la copia y se actualiza el valor DATALINK para señalar la nueva copia del archivo.

RECOVERY

Especifica si DB2 va a dar soporte a la recuperación puntual en el tiempo de los archivos referidos por los valores de esta columna.

YES

DB2 va a dar soporte a la recuperación puntual en el tiempo de los archivos referidos por los valores de esta columna. Sólo puede especificarse este valor cuando también se especifiquen INTEGRITY ALL y WRITE PERMISSION BLOCKED.

CREATE TABLE

NO

Especifica que no se va a dar soporte a la recuperación puntual en el tiempo.

ON UNLINK

Especifica la acción realizada en un archivo cuando se cambia o se suprime un valor DATALINK (desenlace). Tenga en cuenta que esto no se puede aplicar cuando se utiliza WRITE PERMISSION FS.

RESTORE

Especifica que cuando se desenlace un archivo, DataLinks File Manager intentará devolver el archivo al propietario con los permisos que existían en el momento en que se ha enlazado el archivo. Si el usuario ya no está registrado en el servidor de archivos, el resultado es específico de cada producto.
⁷⁷ Esto sólo puede especificarse si también se especifican INTEGRITY ALL y WRITE PERMISSION BLOCKED.

DELETE

Especifica que el archivo se suprimirá cuando se desenlace. Esto sólo puede especificarse cuando también se especifiquen READ PERMISSION DB y WRITE PERMISSION BLOCKED.

MODE DB2OPTIONS

Esta modalidad define un conjunto de opciones de enlace de archivo por omisión. Los valores por omisión definidos por DB2OPTIONS son:

- INTEGRITY ALL
- READ PERMISSION FS
- WRITE PERMISSION FS
- RECOVERY NO

No puede aplicarse ON UNLINK desde el momento en que utiliza WRITE PERMISSION FS.

SCOPE

Identifica el ámbito de la columna de tipo de referencia.

Debe especificarse un ámbito para cualquier columna que vaya a utilizarse como operando izquierdo de un operador de desreferencia o como argumento de la función Deref. La especificación del ámbito de una columna de tipo de referencia

⁷⁷. En DB2 Universal Database, el archivo se asigna a un id de usuario especial predefinido, "dfmunknown".

puede diferirse a una sentencia ALTER TABLE subsiguiente para permitir que se defina la tabla de destino, normalmente en el caso de tablas que se hacen referencia mutuamente.

nombre-tabla-tipo

El nombre de una tabla con tipo. La tabla debe existir ya o ser la misma que el nombre de la tabla que se está creando (SQLSTATE 42704). El tipo de datos de *nombre-columna* debe ser REF(S), donde S es el tipo de *nombre-tabla-tipo* (SQLSTATE 428DM). No se realiza ninguna comprobación de los valores asignados a *nombre-columna* para asegurarse de si realmente los valores hacen referencia a filas existentes de *nombre-tabla-tipo*.

nombre-vista-tipo

El nombre de una vista con tipo. La vista debe existir ya o ser la misma que el nombre de la vista que se está creando (SQLSTATE 42704). El tipo de datos de *nombre-columna* debe ser REF(S), donde S es el tipo de *nombre-vista-tipo* (SQLSTATE 428DM). No se realiza ninguna comprobación de los valores asignados a *nombre-columna* para asegurarse de si realmente los valores hacen referencia a filas existentes de *nombre-vista-tipo*.

CONSTRAINT *nombre-restricción*

Indica el nombre de la restricción. Un *nombre-restricción* no debe identificar a ninguna restricción que ya esté especificada en la misma sentencia CREATE TABLA (SQLSTATE 42710).

Si se omite esta cláusula, se genera un identificador de 18 caracteres, que es exclusivo para cada identificador de las restricciones existentes definidas en la tabla⁷⁸.

Cuando se utiliza con una restricción PRIMARY KEY o UNIQUE, el *nombre-restricción* puede utilizarse como el nombre de un índice que se ha creado para dar soporte a la restricción.

PRIMARY KEY

Proporciona un método abreviado para definir una clave primaria compuesta de una sola columna. De este modo, si se especifica PRIMARY KEY en la definición de la columna C, el efecto es el mismo que si se especifica la cláusula PRIMARY KEY(C) como cláusula separada.

78. El identificador está formado por "SQL" seguido de una secuencia de 15 caracteres numéricos, generados por una función basada en la fecha y la hora.

CREATE TABLE

No puede especificarse una clave primaria si la tabla es una subtabla (SQLSTATE 429B3) porque la clave primaria se hereda de la supertabla.

Consulte el apartado PRIMARY KEY en la descripción de la *restricción-unicidad* más abajo.

UNIQUE

Proporciona un método abreviado de definir una clave de unicidad compuesta de una sola columna. Por lo tanto, si se especifica UNIQUE en la definición de la columna C, el efecto es el mismo que si se especificase la cláusula UNIQUE(C) como una cláusula separada.

No puede especificarse una restricción de unicidad si la tabla es una subtabla (SQLSTATE 429B3) porque las restricciones de unicidad se heredan de la supertabla.

Consulte el apartado UNIQUE en la descripción de la *restricción-unicidad* más abajo.

cláusula-referencias

Proporciona un método abreviado de definir una clave externa compuesta de una sola columna. Así, si se especifica una cláusula-referencias en la definición de la columna C, el efecto es el mismo que si se especificase esa cláusula-referencias como parte de una cláusula FOREIGN KEY en la que C fuera la única columna identificada.

Consulte el apartado *cláusulas-referencias* bajo *restricción-referencia* más abajo.

CHECK (*condición-control*)

Proporciona un método abreviado de definir una restricción de comprobación que se aplica a una sola columna. Vea CHECK (*condición-control*) más adelante.

INLINE LENGTH *entero*

Esta opción sólo es válida para una columna que se ha definido utilizando un tipo estructurado (SQLSTATE 42842) e indica el tamaño máximo, en bytes, de una instancia de un tipo estructurado para su almacenamiento en serie con el resto de valores de la fila. Las instancias de tipos estructurados que no se pueden almacenar en serie se almacenan separadamente de la fila de la tabla base, de forma similar a como se manejan los valores LOB. Esta operación se realiza automáticamente.

El valor por omisión de INLINE LENGTH para una columna de tipo estructurado es la longitud "inline" de su tipo (especificada explícitamente o por omisión en la sentencia CREATE TYPE). Si el

valor `INLINE LENGTH` del tipo estructurado es menor que 292, se utiliza el valor 292 para la columna.

Nota: Las longitudes "inline" de los subtipos no se incluyen en la longitud "inline" por omisión, debido a que las instancias de los subtipos pueden no caber "inline" a menos que, al crear la tabla, se especifique explícitamente un valor `INLINE LENGTH` para tener en cuenta los subtipos actuales y futuros.

El valor `INLINE LENGTH` explícito debe ser igual a 292 como mínimo y no ser mayor que 32672 (SQLSTATE 54010).

espec-predefinida-columna

cláusula-predefinida

Especifica un valor por omisión para la columna.

WITH

Palabra clave opcional.

DEFAULT

Proporciona un valor por omisión en el caso de que no se suministre ningún valor en `INSERT` o se especifique uno como `DEFAULT` en `INSERT` o `UPDATE`. Si no se especifica un valor por omisión a continuación de la palabra clave `DEFAULT`, el valor por omisión depende del tipo de datos de la columna, tal como se muestra en la Tabla 21 en la página 576.

Si una columna se define como `DATALINK`, no puede especificarse un valor por omisión (SQLSTATE 42613). El único valor por omisión posible es `NULL`.

Si la columna está basada en una columna de una tabla con tipo, debe especificarse un valor por omisión específico cuando se defina un valor por omisión. No se puede especificar un valor por omisión para la columna de identificador de objeto de una tabla con tipo (SQLSTATE 42997).

Si se define una columna utilizando un tipo diferenciado, el valor por omisión de la columna es el valor por omisión del tipo de datos fuente convertido al tipo diferenciado.

Si una columna se define utilizando un tipo estructurado, no puede especificarse la *cláusula-predefinida* (SQLSTATE 42842).

CREATE TABLE

La omisión de DEFAULT en una *definición-columna* da como resultado la utilización del valor nulo como valor por omisión para la columna. Si dicha columna está definida como NOT NULL, la columna no tiene un valor por omisión válido.

valores-omisión

Los tipos específicos de los valores por omisión que pueden especificarse son los siguientes.

constante

Especifica la constante como el valor por omisión para la columna. La constante especificada debe:

- representar un valor que pueda asignarse a la columna de acuerdo con las reglas de asignación descritas en el Capítulo 3
- no ser una constante de coma flotante a menos que la columna esté definida con un tipo de datos de coma flotante
- no tener dígitos diferentes de cero fuera de la escala del tipo de datos de la columna si la constante es decimal (por ejemplo, 1,234 no puede ser el valor por omisión para una columna DECIMAL(5,2))
- estar expresada con un máximo de 254 caracteres, incluyendo los caracteres de comillas, cualquier carácter prefijo como la X para una constante hexadecimal, los caracteres del nombre de función completamente calificado y paréntesis, cuando la constante es el argumento de una *función-conversión*.

registro-especial-fechahora

Especifica el valor del registro especial de fecha y hora (CURRENT DATE, CURRENT TIME o CURRENT TIMESTAMP) en el momento de INSERT o UPDATE como valor por omisión para la columna. El tipo de datos de la columna debe ser el tipo de datos que corresponde al registro especial especificado (por ejemplo, el tipo de datos debe ser DATE cuando se especifica CURRENT DATE).

USER

Especifica el valor del registro especial USER en el momento de ejecutar INSERT o UPDATE, como el valor por omisión para la columna. Si se especifica USER, el tipo de datos de la columna debe ser una serie de caracteres con una longitud no inferior al atributo de longitud de USER.

NULL

Especifica NULL como valor por omisión para la columna. Si se ha especificado NOT NULL, puede especificarse DEFAULT NULL en la misma definición de columna, pero se producirá un error si se intenta establecer la columna en el valor por omisión.

función-conversión

Esta forma de valor por omisión sólo puede utilizarse con las columnas definidas como tipo de datos diferenciado, BLOB o de fecha y hora (DATE, TIME o TIMESTAMP). Para el tipo diferenciado, a excepción de los tipos diferenciados basados en tipos BLOB o de fecha y hora, el nombre de la función debe coincidir con el nombre del tipo diferenciado de la columna. Si está calificado con un nombre de esquema, debe ser el mismo que el nombre de esquema del tipo diferenciado. Si no está calificado, el nombre de esquema procedente de la resolución de la función debe ser el mismo que el nombre de esquema del tipo diferenciado. Para un tipo diferenciado basado en un tipo de fecha y hora, en el que el valor por omisión es una constante, debe utilizarse una función y el nombre de ésta debe coincidir con el nombre del tipo fuente del tipo diferenciado, con un nombre de esquema implícito o explícito de SYSIBM. Para las demás columnas de fecha y hora, también puede utilizarse la correspondiente función de fecha y hora. Para un BLOB o tipo diferenciado basado en BLOB, debe utilizarse una función, y el nombre de la función debe ser BLOB, junto con SYSIBM como nombre de esquema implícito o explícito. Para ver un ejemplo de la utilización de la función de conversión, consulte 596.

constante

Especifica una constante como argumento. La constante debe cumplir las reglas de una constante para el tipo fuente del tipo diferenciado, o para el tipo de datos si no se trata de un tipo diferenciado. Si la *función-conversión* es BLOB, la constante debe ser una constante de cadena de caracteres.

registro-especial-fechahora

Especifica CURRENT DATE, CURRENT TIME o CURRENT TIMESTAMP. El tipo fuente del tipo

CREATE TABLE

diferenciado de la columna debe ser el tipo de datos que corresponde al registro especial especificado.

USER

Especifica el registro especial USER. El tipo de datos del tipo fuente del tipo diferenciado de la columna debe ser el tipo de datos serie con una longitud mínima de 8 bytes. Si la *función-conversión* es BLOB, el atributo de longitud debe ser como mínimo 8 bytes.

Si el valor especificado no es válido, se producirá un error (SQLSTATE 42894).

GENERATED

Indica que DB2 genera valores para la columna. Se debe especificar GENERATED si la columna se debe tratar como una columna generada o como columna IDENTITY.

ALWAYS

Indica que DB2 siempre genera un valor para la columna cuando se inserta una fila en la tabla o siempre que pueda cambiar el valor resultante de la *expresión-generación*. El resultado de la expresión se almacena en la tabla. GENERATED ALWAYS es el valor recomendado a menos que se utilice la propagación de datos o se hagan operaciones de descarga y recarga. GENERATED ALWAYS es el valor obligatorio para columnas generadas.

BY DEFAULT

Indica que DB2 genera un valor para la columna cuando se inserta una fila en la tabla, a menos que se especifique un valor. BY DEFAULT es el valor recomendado cuando se utiliza la propagación de datos o se realizan operaciones de descarga/recarga (unload/reload).

Aunque no es explícitamente necesario, es conveniente definir un índice de unicidad para columna generada, a fin de asegurar la unicidad de los valores.

AS IDENTITY

Especifica que la columna debe ser la columna de identidad de la tabla.⁷⁹ Una tabla puede contener una sola columna de identidad (SQLSTATE 428C1). La palabra clave IDENTITY

⁷⁹ Las columnas de identidad no están soportadas en una base de datos con varias particiones (SQLSTATE 42997).

No se puede crear una columna de identidad si la base de datos tiene más de una partición. Si una base de datos contiene alguna columna de identidad, la base de datos no se podrá iniciar con más de una partición.

sólo se puede especificar si el *tipo-datos* de la columna es un tipo numérico exacto⁸⁰ con una escala 0, o un tipo diferenciado, definido por el usuario, para el cual el tipo fuente es un tipo numérico exacto con una escala 0 (SQLSTATE 42815).

Las columnas de identidad están definidas implícitamente como no nulas (NOT NULL).

START WITH *constante-numérica*

Especifica el primer valor de la columna de identidad. Este valor puede ser cualquier valor positivo o negativo que se pueda asignar a esta columna (SQLSTATE 42820), siempre que no existan dígitos distintos de 0 a la derecha de la coma decimal (SQLSTATE 42894). El valor por omisión es 1.

INCREMENT BY *constante-numérica*

Especifica el intervalo existente entre valores consecutivos de la columna de identidad. Este valor puede ser cualquier valor positivo o negativo que se pueda asignar a esta columna (SQLSTATE 42820). Este valor no puede ser cero ni ser mayor que el valor de una constante de entero grande (SQLSTATE 428125), siempre que no existan dígitos distintos de 0 a la derecha de la coma decimal (SQLSTATE 42894).

Si este valor es negativo, se trata de una secuencia descendente. Si este valor es 0 o el valor absoluto es mayor que MAXVALUE - MINVALUE o positivo, se trata de una secuencia ascendente. El valor por omisión es 1.

CACHE o NO CACHE

Especifica si deben mantenerse en la memoria algunos valores pre-asignados, a fin de conseguir un acceso más rápido. Si es necesario un nuevo valor para la columna de identidad y no hay ninguno disponible en la memoria intermedia, entonces el final del nuevo bloque de memoria intermedia se debe anotar en el archivo de anotaciones. En cambio, si es necesario un nuevo valor para la columna de identidad y hay un valor no utilizado en la memoria intermedia, la asignación de ese valor de identidad es más rápida, pues no es necesario hacer

80. Se considera que son tipos numéricos exactos SMALLINT, INTEGER, BIGINT o DECIMAL con una escala 0, o un tipo diferenciado basado en uno de estos tipos. En cambio, las comas flotantes de precisión simple y doble se consideran tipos de datos numéricos aproximados. Los tipos de referencia, aunque estén representados por un tipo numérico exacto, no se pueden definir como columnas de identidad.

CREATE TABLE

ninguna anotación en el archivo de anotaciones. Esta opción se utiliza para el rendimiento y el ajuste.

CACHE *constante-entera*

Especifica cuántos valores de la secuencia de identidad son preasignados y mantenidos en la memoria por DB2. La preasignación y almacenamiento de valores en la memoria intermedia disminuye el registro de anotaciones cuando se generan valores para la columna de identidad.

Si es necesario un nuevo valor para la columna de identidad y no hay ninguno disponible en la memoria intermedia, la asignación del valor requiere que se realice una anotación en el archivo de anotaciones. En cambio, si es necesario un nuevo valor para la columna de identidad y hay un valor no utilizado en la memoria intermedia, la asignación de ese valor de identidad es más rápida, pues no es necesario hacer ninguna anotación en el archivo de anotaciones.

En el caso de que se produzca una desactivación de la base de datos, ya sea de forma normal⁸¹ ya sea debido a una anomalía del sistema, se pierden todos los valores de secuencia almacenados en antememoria que se no han utilizado en sentencias confirmadas. El valor especificado para la opción CACHE es el número máximo de valores de la columna de identidad que se podrían perder en el caso de una desactivación de la base de datos.

El valor mínimo es 2 y el valor máximo es 32767 (SQLSTATE 42815). El valor por omisión es CACHE 20.

NO CACHE

Especifica que no se deben preasignar los valores de la columna de identidad.

Si se especifica esta opción, los valores de la columna de identidad no se colocan en la memoria intermedia. En este caso, cada vez que se solicita un nuevo valor de identidad, se produce una anotación en el archivo de anotaciones.

81. Si una base de datos no se activa explícitamente (utilizando la API o el mandato ACTIVATE), se produce una desactivación implícita cuando se desconecta de la base de datos la última aplicación.

AS (*expresión-generación*)

Especifica que la definición de la columna está basada en una expresión.⁸² La *expresión-generación* no puede contener ninguno de los elementos siguientes (SQLSTATE 42621):

- subconsultas
- funciones de columna
- operaciones de desreferencia o funciones Deref
- funciones no deterministas definidas por el usuario o incorporadas
- funciones definidas por el usuario mediante la opción EXTERNAL ACTION
- funciones definidas por el usuario mediante la opción SCRATCHPAD
- funciones definidas por el usuario mediante la opción READS SQL DATA
- variables del lenguaje principal o marcadores de parámetros
- registros especiales
- referencias a columnas que aparecen definidas más adelante en la lista de columnas
- referencias a otras columnas generadas

El tipo de datos de la columna está basado en el tipo de datos resultante de la *expresión-generación*. Se puede utilizar una especificación CAST para forzar un tipo de datos determinado y proporcionar un ámbito (para un tipo de referencia solamente). Si se especifica *tipo-datos*, se asignan valores a la columna, de acuerdo con las reglas de asignación descritas en “Capítulo 3. Elementos del lenguaje” en la página 69.

Implícitamente se considera que una columna generada puede contener nulos, a menos que se especifique la opción NOT NULL para columnas. El tipo de datos de una columna generada debe ser un tipo para el que esté definida la función de igualdad. Esto excluye las columnas cuyo tipo de datos sea LONG VARCHAR, LONG VARGRAPHIC o LOB, los DATALINK, los tipos estructurados y los tipos diferenciados basados en cualquiera de estos tipos (SQLSTATE 42962).

82. Si la expresión correspondiente a una columna definida como GENERATED ALWAYS incluye una función externa definida por el usuario, la modificación del archivo ejecutable de la función (por ejemplo, cambio de los resultados correspondientes a unos argumentos determinados) puede originar datos incoherentes. Esto se puede evitar utilizando la sentencia SET INTEGRITY para forzar la generación de nuevos valores.

CREATE TABLE

definición-columna-OID

Define la columna de identificador de objeto para la tabla con tipo.

REF IS *nombre-columna-OID* **USER GENERATED**

Especifica que en la tabla se define una columna de identificador de objeto (OID) como la primera columna. Se necesita un OID para la tabla raíz de una jerarquía de tablas (SQLSTATE 428DX). La tabla debe ser una tabla con tipo (debe estar presente la cláusula OF) que no sea una subtabla (SQLSTATE 42613). El nombre de la columna se define como *nombre-columna-OID* y no puede ser el mismo que el nombre de cualquier atributo del tipo estructurado *nombre-tipo1* (SQLSTATE 42711). La columna se define con el tipo *REF(nombre-tipo1)*, NOT NULL y se genera un índice de unicidad requerido por el sistema (con un nombre de índice por omisión). Esta columna viene referida como la *columna de identificador de objeto* o *columna de OID*. Las palabras clave USER GENERATED indican que el usuario debe proporcionar el valor inicial de la columna de OID cuando inserte una fila. Una vez que se haya insertado una fila, la columna de OID no podrá actualizarse (SQLSTATE 42808).

opciones-with

Define opciones adicionales que se aplican a las columnas de una tabla con tipo.

nombre-columna

Especifica el nombre de la columna para la que se especifican las opciones adicionales. El *nombre-columna* debe corresponder al nombre de una columna de la tabla que no sea además una columna de una supertabla (SQLSTATE 428DJ). Sólo puede aparecer un nombre de columna en una cláusula WITH OPTIONS de la sentencia (SQLSTATE 42613).

Si ya está especificada una opción como parte de la definición de tipo (en CREATE TYPE), las opciones especificadas aquí alteran temporalmente las opciones de CREATE TYPE.

WITH OPTIONS *opciones-columna*

Define opciones para la columna especificada. Consulte el valor *opciones-columna* descrito anteriormente. Si la tabla es una subtabla, no pueden especificarse restricciones de unicidad ni de clave primaria (SQLSTATE 429B3).

DATA CAPTURE

Indica si se ha de anotar en la anotación cronológica información

adicional para la duplicación de datos entre bases de datos. No puede especificarse esta cláusula cuando se crea una subtabla (SQLSTATE 42613).

Si la tabla es una tabla con tipo, entonces esta opción no se soporta (SQLSTATE 428DH o 42HDR).

NONE

Indica que no se va a anotar ninguna información adicional.

CHANGES

Indica que en la anotación cronológica se anotará información adicional referente a los cambios de SQL efectuados en esta tabla. Esta opción es necesaria para duplicar la tabla y cuando se utiliza el programa Capture para capturar los cambios contenidos en el archivo de anotaciones para esta tabla.

Si la tabla está definida para permitir datos en una partición que no es la partición del catálogo (grupo de nodos de múltiples particiones o grupo de nodos con una partición que no es la del catálogo), no se da soporte a esta opción (SQLSTATE 42997).

Si el nombre de esquema (implícito o explícito) de la tabla es más largo que 18 bytes, entonces no se da soporte a esta opción (SQLSTATE 42997).

Se puede encontrar más información cerca de la duplicación en el manual *Administration Guide* y en el manual *Replication Guide and Reference*.

IN nombre-espacio-tablas1

Identifica el espacio de tablas en que se va a crear la tabla. El espacio de tablas debe existir y ser un espacio de tablas REGULAR. Si no se especifica ningún otro espacio de tablas, todas las partes de la tabla se almacenarán en este espacio de tablas. No puede especificarse esta cláusula cuando se crea una subtabla (SQLSTATE 42613) porque el espacio de tablas se hereda de la tabla raíz de la jerarquía de tablas. Si no se especifica esta cláusula, el espacio de la tabla se determina de la manera siguiente:

```
IF (Si) existe el espacio de tabla IBMDEFAULTGROUP sobre el que el
  usuario tiene privilegio (de uso) USE
  con un tamaño de página suficiente
  THEN elíjalo
ELSE IF (De lo contrario, si) existe un espacio de tabla sobre el
  que el usuario tiene el privilegio (de uso) USE
  con un tamaño de página suficiente
  (vea más abajo cuándo son idóneos múltiples espacios de tabla)
  THEN elíjalo
ELSE (De lo contrario) emita un error (SQLSTATE 42727).
```

CREATE TABLE

Si la condición ELSE IF identifica más de un espacio de tablas, entonces elija el espacio de tablas con el tamaño de página suficiente más pequeño, para el cual el ID de autorización de la sentencia tenga el privilegio USE. Cuando existe más de un espacio de tablas que puede elegirse, se establece una prioridad basada en quién recibió el privilegio USE:

1. el ID de autorización
2. un grupo al cual pertenece el ID de autorización
3. PUBLIC

Si todavía existe más de un espacio de tablas que puede elegirse, el gestor de bases de datos toma la decisión final.

La determinación del espacio de tablas puede cambiar cuando:

- se eliminan o crean espacios de tablas
- se otorgan o revocan privilegios USE.

El tamaño de página suficiente de una tabla está determinado por el número de bytes de la fila o por el número de columnas. Vea “Tamaño de fila” en la página 885 para obtener más información.

opciones-espaciotablas:

Especifica el espacio tabla en que se almacenarán los valores de los índices y/o de las columnas largas. Consulte el apartado “CREATE TABLESPACE” en la página 893 para obtener detalles acerca de los tipos de espacios de tablas.

INDEX IN *nombre-espacio-tablas2*

Identifica el espacio de tablas donde se crearán todos los índices de la tabla. Esta opción sólo es válida cuando el espacio de tablas principal que se especifica en la cláusula IN sea un espacio de tablas DMS. El espacio de tablas especificado debe existir, debe ser un espacio de tablas DMS regular, para el cual el ID de autorización de la sentencia tiene el privilegio USE, y debe estar en el mismo grupo de nodos que el *nombre-espacio-tablas1* (SQLSTATE 42838).

Observe que la especificación de qué espacio de tablas contendrá el índice de una tabla sólo puede hacerse al crear la tabla. La comprobación del privilegio USE para el espacio de tablas de índice sólo se realiza al crear la tabla. El gestor de bases de datos no exigirá que el ID de autorización de una sentencia CREATE INDEX tenga el privilegio USE para el espacio de tablas cuando se cree un índice más tarde.

LONG IN *nombre-espacio-tablas3*

Identifica el espacio de tablas donde se almacenarán los

valores de todas las columnas largas (tipos de datos LONG VARCHAR, LONG VARCHARIC y LOB, tipos diferenciados basados en cualquiera de esos tipos o columnas definidas con tipos estructurados definidos por el usuario que contienen valores que no se pueden almacenar internamente ("inline"). Esta opción sólo es válida cuando el espacio de tablas principal que se especifica en la cláusula IN sea un espacio de tablas DMS. El espacio de tablas debe existir, debe ser un espacio de tablas DMS largo (LONG), para el cual el ID de autorización de la sentencia tenga el privilegio USE, y debe estar en el mismo grupo de nodos que *nombre-espacio-tablas1* (SQLSTATE 42838).

Observe que la especificación de qué espacio de tablas contendrá las columnas largas y de tipo LOB de una tabla sólo puede hacerse al crear la tabla. La comprobación del privilegio USE para el espacio de tablas que contendrá las columnas largas y de tipo LOB sólo se realiza al crear la tabla. El gestor de bases de datos no exigirá que el ID de autorización de una sentencia ALTER TABLE tenga el privilegio USE para el espacio de tablas cuando más tarde se añada una columna larga o de tipo LOB.

PARTITIONING KEY (*nombre-columna,...*)

Especifica la clave de particionamiento utilizada cuando se particionan los datos de la tabla. Cada *nombre-columna* debe identificar una columna de la tabla y la misma columna no debe estar identificada más de una vez. Una columna no se puede utilizar como parte de una clave de particionamiento si el tipo de datos de la columna es un LONG VARCHAR, LONG VARCHARIC, BLOB, CLOB, DBCLOB, DATALINK, un tipo diferenciado basado en cualquiera de estos tipos o un tipo estructurado (SQLSTATE 42962). No puede especificarse una clave de particionamiento para una tabla que sea una subtabla (SQLSTATE 42613) porque la clave de particionamiento se hereda de la tabla raíz de la jerarquía de tablas.

Si no se especifica esta cláusula y esta tabla reside en un grupo de nodos de múltiples particiones, la clave de particionamiento se define de la manera siguiente:

- si la tabla es una tabla con tipo, la columna de identificador de objeto es la clave de particionamiento
- si se especifica una clave primaria, la primera columna de la clave primaria es la clave de particionamiento
- en otros caso, la clave de particionamiento es la primera columna cuyo tipo de datos no sea una columna LOB, LONG

CREATE TABLE

VARCHAR, LONG VARCHAR ni DATALINK, un tipo diferenciado basado en uno de estos tipos ni una columna de tipo estructurado.

Si ninguna de las columnas satisface el requisito de la clave de particionamiento por omisión, la tabla se crea sin ninguna. Dichas tablas sólo están permitidas en los espacios de tablas definidos en los grupos de nodos de una sola partición.

Para las tablas de los espacios de tablas definidos en los grupos de nodos de una sola partición, puede utilizarse cualquier conjunto de columnas que no sean de tipo largo para definir la clave de particionamiento. Si no especifica este parámetro, no se crea ninguna clave de particionamiento.

Para ver las restricciones relacionadas con la clave de particionamiento, consulte el apartado “Normas” en la página 879.

USING HASHING

Especifica la utilización de la función de generación aleatoria como el método de partición para la distribución de datos. Es el único método de partición soportado.

REPLICATED

Especifica que los datos almacenados en la tabla se duplican físicamente en cada partición de base de datos del grupo de nodos del espacio de tablas en que está definida la tabla. Esto significa que existe una copia de todos los datos de la tabla en cada una de estas particiones de base de datos. Esta opción sólo puede especificarse para una tabla de resumen (SQLSTATE 42997).

NOT LOGGED INITIALLY

Los cambios realizados en la tabla por una operación de Inserción, Supresión, Actualización, Creación de índice, Eliminación de índice o Modificación de tabla durante la misma unidad de trabajo en la que se crea la tabla no se registran en el archivo de anotaciones. Consulte el apartado “Notas” en la página 881 para ver otras consideraciones a tener en cuenta al utilizar esta opción.

Todos los cambios de catálogo e información relativa al almacenamiento se anotan cronológicamente, así como las todas las operaciones que se realizan en la tabla en unidades de trabajo subsiguientes.

No puede definirse una restricción de clave foránea en una tabla que haga referencia a un padre con el atributo NOT LOGGED INITIALLY. No puede especificarse esta cláusula cuando se crea una subtabla (SQLSTATE 42613).

Nota: En la misma unidad de trabajo donde se crea una tabla definida como NOT LOGGED INITIALLY no se puede emitir una petición para retrotraer hasta un punto de salvar. Esto originaría un error (SQLSTATE 40506) y se produciría una retrotracción de la unidad de trabajo completa.

restricción-unicidad

Define una restricción de clave primaria o de unicidad. Si la tabla tiene una clave de particionamiento, cualquier clave de unicidad o primaria debe ser un superconjunto de la clave de particionamiento. No puede especificarse una restricción de unicidad o de clave primaria para una tabla que sea una subtabla (SQLSTATE 429B3). Si la tabla es una tabla raíz, la restricción se aplica a la tabla y a todas sus subtablas.

CONSTRAINT *nombre-restricción*

Designa la restricción de clave primaria o restricción de unicidad. Vea la página 859.

UNIQUE (*nombre-columna,...*)

Define una clave de unicidad compuesta por las columnas identificadas. Las columnas identificadas deben estar definidas como NOT NULL. Cada *nombre-columna* debe identificar una columna de la tabla y la misma columna no debe estar identificada más de una vez.

El número de columnas identificadas no debe exceder de 16 y la suma de sus longitudes almacenadas no debe exceder de 1024 (consulte el apartado "Cuentas de bytes" en la página 885 para conocer las longitudes almacenadas). La longitud de una columna individual cualquiera no debe exceder de 255 bytes. Esta longitud sólo es para los datos y no ve afectada por el byte nulo, si existiera. La longitud máxima de los datos de una columna es 255 bytes, tanto si la columna puede contener nulos como si no. La clave de unicidad no puede contener ningún LOB, LONG VARCHAR, LONG VARCHARIC, DATALINK, un tipo diferenciado basado en cualquiera de estos tipos ni un tipo estructurado, aunque el atributo de longitud de la columna sea lo bastante pequeño como para caber dentro del límite de 255 bytes (SQLSTATE 42962).

El conjunto de columnas de la clave de unicidad no puede ser el mismo que el conjunto de columnas de la clave primaria o de otra clave de unicidad (SQLSTATE 01543).⁸³

No puede especificarse una restricción de unicidad si la tabla es una subtabla (SQLSTATE 429B3) porque las restricciones de unicidad se heredan de la supertabla.

83. Si LANGLEVEL es SQL92E o MIA, se devuelve un error, SQLSTATE 42891.

CREATE TABLE

La descripción de la tabla, tal como está registrada en el catálogo, incluye la clave de unicidad y su índice de unicidad. Se creará automáticamente un índice de unicidad para las columnas en la secuencia especificada por orden ascendente para cada columna. El nombre del índice será el mismo que el *nombre-restricción* si no entra en conflicto con un índice existente en el esquema donde se ha creado la tabla. Si el nombre del índice entra en conflicto, el nombre será SQL, seguido de una indicación de la hora de caracteres (*aammddhhmmssxxx*), con SYSIBM como el nombre de esquema.

PRIMARY KEY (*nombre-columna,...*)

Define una clave primaria formada por las columnas identificadas. La cláusula no debe especificarse más de una vez y las columnas identificadas deben estar definidas como NOT NULL. Cada *nombre-columna* debe identificar una columna de la tabla y la misma columna no debe estar identificada más de una vez.

El número de columnas identificadas no debe exceder de 16 y la suma de sus longitudes almacenadas no debe exceder de 1024 (consulte el apartado "Cuentas de bytes" en la página 885 para conocer las longitudes almacenadas). La longitud de una columna individual cualquiera no debe exceder de 255 bytes. Esta longitud sólo es para los datos y no ve afectada por el byte nulo, si existiera. La longitud máxima de los datos de una columna es 255 bytes, tanto si la columna puede contener nulos como si no. La clave primaria no puede contener ningún LOB, LONG VARCHAR, LONG VARGRAPHIC, DATALINK, un tipo diferenciado basado en uno de estos tipos ni un tipo estructurado, aunque el atributo de longitud de la columna sea lo bastante pequeño como para caber dentro del límite de 255 bytes (SQLSTATE 42962).

El conjunto de columnas en la clave primaria no puede ser el mismo que el conjunto de columnas de una clave de unicidad (SQLSTATE 01543).⁸³

En una tabla sólo se puede definir una clave primaria.

No puede especificarse una clave primaria si la tabla es una subtabla (SQLSTATE 429B3) porque la clave primaria se hereda de la supertabla.

La descripción de la tabla, tal como está registrada en el catálogo, incluye la clave primaria y su índice principal. Se creará automáticamente un índice de unicidad para las columnas en la secuencia especificada por orden ascendente para cada columna. El nombre del índice será el mismo que el *nombre-restricción* si no entra en conflicto con un índice existente en el esquema donde se ha creado la tabla. Si el nombre del índice entra en conflicto, el nombre será

SQL, seguido de una indicación de la hora de caracteres (*aammddhhmmssxxx*), con SYSIBM como el nombre de esquema.

Si la tabla tiene una clave de particionamiento, las columnas de una *restricción-unicidad* deben ser un superconjunto de las columnas de la clave de particionamiento; el orden de las columnas no es importante.

restricción-referencia

Define una restricción de referencia.

CONSTRAINT *nombre-restricción*

Indica el nombre de la restricción de referencia. Véase la página 859.

FOREIGN KEY (*nombre-columna,...*)

Define una restricción de referencia con el *nombre-restricción* especificado.

T1 indicará la tabla de objetos de la sentencia. La clave foránea de la restricción de referencia se compone de las columnas identificadas. Cada nombre de la lista de nombres de columna debe identificar una columna de T1, y no debe identificarse la misma columna en más de una ocasión. El número de columnas identificadas no debe exceder de 16 y la suma de sus longitudes almacenadas no debe exceder de 1024 (consulte el apartado "Cuentas de bytes" en la página 885 para conocer las longitudes almacenadas). La clave foránea no puede contener ningún LOB, LONG VARCHAR, LONG VARGRAPHIC, DATALINK, un tipo diferenciado basado en uno de estos tipos ni una columna de tipo estructurado (SQLSTATE 42962). Debe haber el mismo número de columnas de clave foránea que hay en la clave padre y los tipos de datos de las columnas correspondientes deben ser compatibles (SQLSTATE 42830). Dos descripciones de columna son compatibles si tienen tipos de datos compatibles (ambas columnas son numéricas, de series de caracteres, gráficas, fecha/hora o tienen el mismo tipo diferenciado).

cláusula-referencias

Especifica la tabla padre y la clave padre para la restricción de referencia.

REFERENCES *nombre-tabla*

La tabla especificada en una cláusula REFERENCES debe identificar una tabla base que esté descrita en el catálogo, pero no debe identificar una tabla de catálogo.

Una restricción de referencia es un duplicado si su clave foránea, clave padre y tabla padre son iguales que la clave foránea, clave padre y tabla padre de una restricción de referencia especificada previamente. Las restricciones de referencias duplicadas se ignoran y se emite un aviso (SQLSTATE 01543).

CREATE TABLE

En la siguiente explicación, T2 indicará la tabla padre identificada y T1 indicará la tabla que se está creando ⁸⁴ (T1 y T2 pueden ser la misma tabla).

La clave foránea especificada debe tener el mismo número de columnas que la clave padre de T2 y la descripción de la columna *n* de la clave foránea debe ser comparable a la descripción de la columna *n* de dicha clave padre. Las columnas de fecha y hora no se consideran compatibles con las columnas de serie al aplicar esta regla.

(nombre-columna,...)

La clave padre de la restricción de referencia se compone de las columnas identificadas. Cada *nombre-columna* debe ser un nombre no calificado que identifique una columna de T2. La misma columna no se puede identificar más de una vez.

La lista de nombres de columna debe coincidir con el conjunto de columnas (en cualquier orden) de la clave primaria o con una restricción de unicidad que exista en T2 (SQLSTATE 42890). Si no se especifica una lista de nombres de columna, T2 debe tener una clave primaria (SQLSTATE 42888). La omisión de la lista de nombres de columna es una especificación implícita de las columnas de dicha clave primaria en la secuencia especificada originalmente.

La restricción de referencia especificada por una cláusula FOREIGN KEY define una relación en la que T2 es padre y T1 es dependiente.

cláusula-regla

Especifica la acción que debe realizarse en las tablas dependientes.

ON DELETE

Especifica la acción que se ha de emprender en las tablas dependientes al suprimir una fila de la tabla superior. Existen cuatro acciones posibles:

- NO ACTION (valor por omisión)
- RESTRICT
- CASCADE
- SET NULL

84. o modificando, en caso de que se haga referencia a esta cláusula en la descripción de la sentencia ALTER TABLE.

La regla de supresión se aplica cuando una fila de la T2 es el objeto de una operación de DELETE o de supresión propagada y esa fila tiene dependientes en la T1. Supongamos que p indica dicha fila de T2.

- Si se especifica RESTRICT o NO ACTION, se produce un error y no se suprime ninguna fila.
- Si se especifica CASCADE, la operación de supresión se propaga a los dependientes de p en T1.
- Si se especifica SET NULL, cada columna con posibilidad de nulos de la clave foránea de cada dependiente de p en T1 se establece en nulo.

No debe especificarse SET NULL a menos que alguna columna de las claves foráneas permita valores nulos. La omisión de esta cláusula es una especificación implícita de ON DELETE NO ACTION.

Un ciclo que implica dos o más tablas no debe hacer que una tabla se conecte para supresión consigo misma a menos que todas las reglas de supresión del ciclo sean CASCADE. De esta forma, si la nueva relación va a formar un ciclo y T2 ya está conectada para supresión con T1, entonces la restricción sólo puede definirse si tiene una regla de supresión de CASCADE y todas las demás reglas de supresión del ciclo son CASCADE.

Si T1 está conectada para supresión a T2 a través de múltiples vías de acceso, las relaciones en las que T1 sea dependiente y que compongan, en todo o en parte, esas vías deben tener la misma regla de supresión y ésta no debe ser SET NULL. Las acciones NO ACTION y RESTRICT se tratan de manera idéntica. Por lo tanto, si T1 es dependiente de T3 en una relación con una regla de supresión r , la restricción de referencia no puede definirse cuando r se establece en SET NULL si existe alguna de estas condiciones:

- T2 y T3 son la misma tabla
- T2 es descendiente de T3 y la supresión de filas se realiza en cascada de T3 a T2
- T3 es descendiente de T2 y la supresión de filas se realiza en cascada de T2 a T3
- T2 y T3 son ambas descendientes de una misma tabla y la supresión de filas de esa tabla se realiza en cascada a T2 y T3.

CREATE TABLE

Si *r* no es SET NULL, puede definirse la restricción de referencia, pero la regla de supresión que se especifica implícita o explícitamente en la cláusula FOREIGN KEY debe ser igual que *r*.

Cuando se aplican las reglas anteriores a restricciones de referencia, en las que la tabla padre o la tabla dependiente es un miembro de una jerarquía de tablas con tipo, se tienen en cuenta todas las restricciones de referencia aplicables a cualquier tabla de sus respectivas jerarquías.

ON UPDATE

Especifica la acción que se ha de emprender en las tablas dependientes al actualizar una fila de la tabla padre. La cláusula es opcional. ON UPDATE NO ACTION es el valor por omisión y ON UPDATE RESTRICT es la única alternativa.

La diferencia entre NO ACTION y RESTRICT se describe bajo CREATE TABLE en el apartado “Notas” en la página 881.

restricción-comprobación

Define una restricción de comprobación. Una *restricción-comprobación* es una *condición-búsqueda* que debe evaluarse a no falsa.

CONSTRAINT *nombre-restricción*

Indica el nombre de la restricción de comprobación. Vea la página 859.

CHECK (*condición-control*)

Define una restricción de comprobación. Una *condición-control* es una *condición-búsqueda*, pero con las salvedades siguientes:

- Una referencia a columna debe serlo a una columna de la tabla que se está creando
- La *condición-búsqueda* no puede contener un predicado TYPE
- No puede contener ninguno de los elementos siguientes (SQLSTATE 42621):
 - subconsultas
 - operaciones de desreferencia o funciones Deref donde el argumento de referencia con ámbito es distinto de la columna de identificador de objeto (OID).
 - especificaciones CAST con una cláusula SCOPE
 - funciones de columna
 - funciones que no sean deterministas
 - funciones definidas para tener una acción externa
 - funciones definidas por el usuario mediante la opción SCRATCHPAD

- funciones definidas por el usuario mediante la opción READS SQL DATA
- variables del sistema principal
- marcadores de parámetros
- registros especiales
- un seudónimo
- referencias a columnas generadas distintas de la columna de identidad

Si se especifica una restricción de comprobación como parte de una *definición-columna*, sólo puede hacerse una referencia de columna para la misma columna. Las restricciones de comprobación especificadas como parte de una definición de tabla pueden tener referencias de columna que identifiquen columnas que ya hayan sido definidas previamente en la sentencia CREATE TABLE. En las restricciones de comprobación no se verifica si hay incoherencias, duplicados o condiciones equivalentes. Por este motivo, se pueden definir restricciones de comprobación que sean contradictorias o redundantes, lo que puede dar lugar a posibles errores en tiempo de ejecución.

Existe la posibilidad de especificar la condición de control "IS NOT NULL", si bien es recomendable que la posibilidad de contener nulos se imponga directamente utilizando el atributo NOT NULL de las columnas. Por ejemplo, CHECK (salario + bonificación > 30000) se acepta si el salario se establece en NULL, porque las condiciones de CHECK deben cumplirse o bien son desconocidas, en cuyo caso el salario no se conoce. En cambio, CHECK (salario IS NOT NULL) se consideraría falso y, por lo tanto, una violación de la restricción si el salario se establece en NULL.

Las restricciones de comprobación se imponen al insertar o actualizar filas de la tabla. Una restricción de comprobación definida en una tabla se aplica automáticamente a todas las subtablas de esta tabla.

Normas

- La suma de las cuentas de bytes de las columnas, incluidas las longitudes "inline" de todas las columnas de tipo estructurado, no debe ser mayor que el límite de tamaño de la fila, que está basado en el tamaño de página del espacio de tablas (SQLSTATE 54010). Vea "Cuentas de bytes" en la página 885 y la Tabla 35 en la página 1256 para obtener más información. En las tablas con tipo, el número de bytes se aplica a las columnas de la tabla raíz de la jerarquía de tablas y a cada columna adicional introducida por cada subtabla de la jerarquía de tablas (las columnas adicionales de las subtablas deben considerarse con posibilidad de nulos para los fines del

CREATE TABLE

número de bytes aunque estén definidas sin posibilidad de nulos). También hay 4 bytes adicionales de actividad general para identificar la subtabla a la que pertenece cada fila.

- El número de columnas en una tabla no puede sobrepasar la cantidad de 1.012 (SQLSTATE 54011). Para las tablas con tipo, el número total de atributos de los tipos de todas las subtablas de la jerarquía de tablas no puede sobrepasar la cantidad de 1010.
- Una columna de identificador de objeto de una tabla con tipo no puede actualizarse (SQLSTATE 42808).
- Una columna de clave de particionamiento puede actualizarse a no ser que el parámetro de configuración DB2_UPDATE_PART_KEY esté establecido en 'OFF' (SQLSTATE 42997). 'OFF' es el valor por omisión.
- Cualquier clave de unicidad o primaria definida en la tabla debe ser un superconjunto de la clave de particionamiento (SQLSTATE 42997).
- Una columna que permita nulos de una clave de particionamiento puede incluirse como columna de clave foránea cuando se define la relación con ON DELETE SET NULL, a no ser que el parámetro de configuración DB2_UPDATE_PART_KEY se haya establecido en 'OFF' (SQLSTATE 42997).
- La tabla siguiente proporciona las combinaciones soportadas de opciones DATALINK dentro de las *opciones-enlace-archivo* (SQLSTATE 42613).

Tabla 24. Combinaciones válidas de opciones de control de archivo DATALINK

INTEGRITY	READ PERMISSION	WRITE PERMISSION	RECOVERY	ON UNLINK
ALL	FS	FS	NO	No aplicable
ALL	FS	BLOCKED	NO	RESTORE
ALL	FS	BLOCKED	YES	RESTORE
ALL	DB	BLOCKED	NO	RESTORE
ALL	DB	BLOCKED	NO	DELETE
ALL	DB	BLOCKED	YES	RESTORE
ALL	DB	BLOCKED	YES	DELETE

Las reglas siguientes sólo se aplican a las bases de datos particionadas.

- Las tablas formadas sólo de columnas de los tipos LOB, LONG VARCHAR, LONG VARGRAPHIC, DATALINK, tipos diferenciados basados en uno de estos tipos o de tipo estructurado sólo se pueden crear en espacios de tablas definidos en grupos de nodos de partición única.
- La definición de clave de particionamiento de una tabla de un espacio de tablas definido en un grupo de nodos de múltiples particiones no se puede modificar.

- La columna de clave de particionamiento de una tabla con tipo debe ser la columna de OID.

Notas

- La creación de una tabla con un nombre de esquema que todavía no existe dará como resultado la creación implícita de dicho esquema siempre que el ID de autorización de la sentencia tenga la autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN en el esquema se otorga a PUBLIC.
- Si se especifica una clave foránea:
 - Se invalidan todos los paquetes que estén sujetos a supresión en la tabla padre.
 - Se invalidan todos los paquetes que estén sujetos a actualización en una columna como mínimo de la clave padre.
- La creación de una subtabla causa la invalidación de todos los paquetes que dependan de cualquier tabla de la jerarquía de tablas.
- Columnas VARCHAR y VARGRAPHIC que son mayores de 4.000 y 2.000 respectivamente no deben utilizarse como parámetros de entrada en funciones en esquema SYSFUN. Se producirán errores cuando se invoca la función con un valor de argumento que sobrepasa estas longitudes (SQLSTATE 22001).
- La utilización de NO ACTION o RESTRICT como reglas de supresión o actualización para restricciones de referencia determina cuándo se aplica la restricción. Una regla de supresión o actualización RESTRICT se aplica antes de otras restricciones, incluyendo las restricciones de referencia con reglas de modificación como por ejemplo CASCADE o SET NULL. Una regla de supresión o actualización NO ACTION se aplica después de otras restricciones de referencia. Existen muy pocos casos en que esto pueda suponer una diferencia durante una supresión o actualización. Un ejemplo en el que es evidente un funcionamiento distinto implica la ejecución de DELETE en filas en una vista que está definida como UNION ALL de tablas relacionadas.

La tabla T1 es una tabla padre de la tabla T3, regla de supresión que se explica a continuación

La tabla T2 es una tabla padre de la tabla T3, regla de supresión CASCADE

```
CREATE VIEW V1 AS SELECT * FROM T1 UNION ALL SELECT * FROM T2
```

```
DELETE FROM V1
```

Si la tabla T1 es una tabla padre de la tabla T3 con la regla de supresión RESTRICT, se generará una violación de restricción (SQLSTATE 23001) si existen filas hijos para las claves padre de T1 en T3.

CREATE TABLE

Si la tabla T1 es una tabla padre de la tabla T3 con la regla de supresión NO ACTION, las filas hijos pueden suprimirse mediante la regla de supresión CASCADE al suprimir las filas de la tabla T2, antes de que la regla de supresión NO ACTION se aplique para las supresiones de T1. Si las supresiones de T2 no han tenido como resultado la supresión de todas las filas hijos para las claves padre de T1 en T3, se generará una violación de restricción (SQLSTATE 23504).

Observe que el SQLSTATE que se devuelve es diferente dependiendo de si la regla de supresión o actualización es RESTRICT o NO ACTION.

- Para las tablas situadas en espacios de tablas definidos en grupos de nodos de múltiples particiones, debe tenerse en cuenta la colocación de la tabla al elegir las claves de particionamiento. A continuación encontrará una lista de elementos a tener en cuenta:
 - Las tablas deben estar en el mismo grupo de nodos para la colocación. Los espacios de tablas pueden ser diferentes, pero deben definirse en el mismo grupo de nodos.
 - Las claves de particionamiento de las tablas deben tener el mismo número de columnas y las columnas de clave correspondientes deben tener particiones compatibles para la colocación. Para obtener más información, vea “Compatibilidad entre particiones” en la página 127.
 - La elección de la clave de particionamiento también afecta al rendimiento de las uniones. Si una tabla se une con frecuencia a otra tabla, debe tomar en consideración la posibilidad de unir la columna o columnas como una clave de particionamiento para ambas tablas.
- La cláusula NOT LOGGED INITIALLY no puede utilizarse cuando las columnas DATALINK con el atributo FILE LINK CONTROL están presentes en la tabla (SQLSTATE 42613).
- La opción NOT LOGGED INITIALLY es útil para las situaciones en que se ha de crear un conjunto resultante grande con datos de una fuente alternativa (otra tabla o un archivo) y la recuperación de la tabla no es necesaria. La utilización de esta opción ahorrará la actividad general de anotar cronológicamente los datos. Las siguientes consideraciones se aplican cuando se especifica esta opción:
 - Cuando se confirma la unidad de trabajo, todos los cambios que se han realizado en la tabla durante la unidad de trabajo fluyen al disco.
 - Cuando se ejecuta el programa de utilidad Rollforward (Avanzar) y encuentra un registro de anotación cronológica que indica que una tabla de la base de datos se ha llenado mediante el programa de utilidad Load (Carga) o se ha creado con la opción NOT LOGGED INITIALLY, la tabla se marcará como no disponible. El programa de utilidad Rollforward (Avanzar) eliminará la tabla si, posteriormente, encuentra una anotación cronológica DROP TABLE. De lo contrario, después de recuperar la base

de datos, se emitirá un error si se realiza un intento de acceder a la tabla (SQLSTATE 55019). La única operación permitida es eliminar la tabla.

- Cuando se ha hecho copia de seguridad de la tabla como parte de una copia de seguridad de la base de datos o del espacio de tablas, se puede recuperar la tabla.
- Si CURRENT REFRESH AGE se establece en ANY, se puede utilizar una tabla de resumen REFRESH DEFERRED, definida con ENABLE QUERY OPTIMIZATION, para optimizar el proceso de las consultas. Una tabla de resumen REFRESH IMMEDIATE, definida con ENABLE QUERY OPTIMIZATION, es siempre elegible para la optimización. Para que esta optimización pueda utilizar una tabla de resumen REFRESH DEFERRED o REFRESH IMMEDIATE, la selección completa debe cumplir ciertas reglas además de aquéllas ya descritas. La selección completa:
 - debe ser una subselección con una cláusula GROUP BY o con una sola tabla de referencia
 - no debe incluir DISTINCT en ninguna parte dentro de la lista de selección
 - no debe incluir registros especiales
 - no debe incluir funciones que no sean deterministas.

Si la consulta especificada al crear una tabla de resumen REFRESH DEFERRED no se ajusta a estas reglas, se devuelve un aviso (SQLSTATE 01633).

- Si se define una tabla de resumen con REFRESH IMMEDIATE, es posible que se produzca un error al intentar aplicar el cambio que resulta de insertar, actualizar o eliminar una tabla subyacente. El error provocará la anomalía de inserción, actualización o eliminación de la tabla subyacente.
- Se puede definir una restricción de referencia de forma que la tabla padre o la tabla dependiente forme parte de una jerarquía de tablas. En este caso, el efecto de la restricción de referencia es el siguiente:
 1. Efectos de las sentencias INSERT, UPDATE y DELETE:
 - Si existe una restricción de referencia, en la que TP es una tabla padre y TD es una tabla dependiente, la restricción asegura que para cada fila de TD (o de cualquiera de sus subtablas) que tenga una clave foránea no nula, existe una fila en TP (o en una de sus subtablas) con una clave padre coincidente. Esta regla se aplica para cualquier acción que afecte a una fila de TP o TD, con independencia de cómo se inicie la acción.
 2. Efectos sobre las sentencias DROP TABLE:
 - para las restricciones de referencia en las que la tabla eliminada es la tabla padre o la tabla dependiente, se elimina la restricción
 - para las restricciones de referencia en las que la tabla padre es una supertabla de la tabla eliminada, se consideran eliminadas de la

CREATE TABLE

supertabla las filas de la tabla eliminada. Se comprueba el cumplimiento de la restricción de referencia y su regla de supresión se invoca para fila suprimida.

- para las restricciones de referencia en las que la tabla dependiente es una supertabla de la tabla eliminada, no se comprueba el cumplimiento de la restricción. La supresión de una fila de una tabla dependiente no puede producir una violación de una restricción de referencia.
- **Tablas de resumen no operativas:** Una tabla de resumen no operativa es una tabla que ya no está disponible para las sentencias de SQL. Una tabla de resumen se vuelve no operativa si:
 - Se revoca un privilegio del que depende la definición de la tabla de resumen.
 - Se elimina un objeto, como, por ejemplo, una tabla, un seudónimo o una función, del que depende la definición de la tabla de resumen.

En otras palabras, una tabla de resumen no operativa es una tabla en la que se ha eliminado involuntariamente la definición de tabla de resumen. Por ejemplo, cuando se elimina un seudónimo, cualquier tabla de resumen definida con ese seudónimo deja de ser operativa. Todos los paquetes dependientes de la tabla de resumen dejan de ser válidos.

Hasta que no se vuelva a crear o a eliminar explícitamente la tabla de resumen no operativa, no podrá compilarse ninguna sentencia que utilice esa tabla de resumen no operativa (SQLSTATE 51024), exceptuando las sentencias CREATE ALIAS, CREATE TABLE, DROP TABLE y COMMENT ON TABLE. Hasta que no se haya eliminado explícitamente la tabla de resumen no operativa, no podrá utilizarse su nombre calificado para crear otra vista, tabla base o seudónimo (SQLSTATE 42710).

Una tabla de resumen no operativa puede volver a crearse emitiendo una sentencia CREATE TABLE que utilice el texto de definición de la tabla de resumen no operativa. Este texto de consulta de tabla de resumen se almacena en la columna TEXT del catálogo SYSCAT.VIEWS. Cuando se vuelve a crear una tabla de resumen no operativa, es necesario otorgar de forma explícita todos los privilegios que otros necesitan en dicha tabla, debido al hecho de que se suprimen todos los registros de autorizaciones en una tabla de resumen si la tabla de resumen se marca como no operativa. Tenga en cuenta que no es necesario eliminar de manera explícita la tabla de resumen no operativa para volverla a crear. La emisión de una sentencia CREATE TABLE que defina una tabla de resumen con el mismo *nombre-tabla* que el de una tabla de resumen no operativa hará que se sustituya esta tabla de resumen no operativa y que la sentencia CREATE TABLE devuelva un aviso (SQLSTATE 01595).

Las tablas de resumen no operativas se indican mediante una X en la columna VALID de la vista de catálogo SYSCAT.VIEWS y mediante una X en la columna STATUS de la vista de catálogo SYSCAT.TABLES.

- **Privilegios:** Cuando se crea una tabla, se otorga el privilegio CONTROL al definidor de la tabla. Cuando se crea una subtabla, el definidor de la tabla otorga automáticamente para la subtabla el privilegio SELECT que cada usuario o grupo tiene sobre la supertabla inmediata.
- **Tamaño de fila:** El número máximo de bytes permitidos en la fila de una tabla depende del tamaño de página del espacio de tablas donde se crea la tabla (*nombre-espacio-tablas1*). La lista siguiente muestra el límite del tamaño de fila y el límite del número de columnas asociados a cada tamaño de página del espacio de tablas.

Tabla 25. Límites para el número de columnas y el tamaño de fila correspondientes a cada tamaño de página del espacio de tablas

Tamaño de página	Límite del tamaño de fila	Límite de la cuenta de columnas
4K	4 005	500
8K	8 101	1 012
16K	16 293	1 012
32K	32 677	1 012

El número de columnas real para una tabla puede limitarse más mediante la fórmula siguiente:

- $\text{Número total de columnas} * 8 + \text{Número de columnas LOB} * 12 + \text{Número de columnas de enlaces de datos} * 28 \leq \text{límite del tamaño de fila para el tamaño de página.}$
- **Cuentas de bytes:** La lista siguiente muestra, para cada tipo de datos, el número de bytes de las columnas que no permiten valores nulos. En el caso de las columnas que sí permiten valores nulos, el número de bytes será uno más del que aparece en la lista.

Si la tabla se crea sobre la base de un tipo estructurado, se reservan 4 bytes adicionales de actividad general para identificar filas de subtablas sin tener en cuenta si se han definido subtablas. Además, las columnas adicionales de las subtablas deben considerarse con posibilidad de nulos para los fines del número de bytes aunque estén definidas sin posibilidad de nulos.

Tipo de datos	Número de bytes
INTEGER	4
SMALLINT	2
BIGINT	8

CREATE TABLE

REAL	4
DOUBLE	8
DECIMAL	Parte entera de $(p/2)+1$, donde p es la precisión.
CHAR(n)	n
VARCHAR(n)	$n+4$
LONG VARCHAR	24
GRAPHIC(n)	$n*2$
VARGRAPHIC(n)	$(n*2)+4$
LONG VARGRAPHIC	24
DATE	4
TIME	3
TIMESTAMP	10
DATALINK(n)	$n+54$

Tipos LOB

Cada valor LOB tiene un *descriptor LOB* en el registro base que apunta a la ubicación del valor real. El tamaño de dicho descriptor varía en función de la longitud máxima que se haya definido para la columna. En la tabla siguiente se muestran los tamaños habituales:

Longitud máx. LOB	Tam. descriptor LOB
1 024	72
8 192	96
65 536	120
524 000	144
4 190 000	168
134 000 000	200
536 000 000	224
1 070 000 000	256
1 470 000 000	280
2 147 483 647	316

Tipo diferenciado

Longitud del tipo fuente del tipo diferenciado.

Tipo de referencia

Longitud del tipo de datos incorporado en el que está basado el tipo de referencia.

Tipo estructurado

El valor `INLINE LENGTH + 4`. `INLINE LENGTH` es el valor especificado (o

calculado implícitamente) para la columna en la cláusula *opciones-columna*.

- También se soporta la sintaxis siguiente: NOMINVALUE, NOMAXVALUE, NOCYCLE, NOCACHE y NOORDER.

Ejemplos

Ejemplo 1: Cree la tabla TDEPT en el espacio de tablas DEPARTX. DEPTNO, DEPTNAME, MGRNO y ADMRDEPT son nombres de columnas. CHAR significa que la columna contendrá datos de caracteres. NOT NULL significa que la columna no puede contener ningún valor nulo. VARCHAR significa que la columna contendrá datos de caracteres de longitud variable. La clave primaria es la columna DEPTNO.

```
CREATE TABLE TDEPT
  (DEPTNO  CHAR(3)      NOT NULL,
   DEPTNAME VARCHAR(36) NOT NULL,
  MGRNO    CHAR(6),
   ADMRDEPT CHAR(3)    NOT NULL,
   PRIMARY KEY(DEPTNO))
IN DEPARTX
```

Ejemplo 2: Cree la tabla PROJ en el espacio de tablas SCHED. PROJNO, PROJNAME, DEPTNO, RESPEMP, PRSTAFF, PRSTDATE, PRENDATE y MAJPROJ son nombres de columnas. CHAR significa que la columna contendrá datos de caracteres. DECIMAL significa que la columna contendrá datos decimales empaquetados. 5,2 significa lo siguiente: 5 indica el número de dígitos decimales y 2 indica el número de dígitos a la derecha de la coma decimal. NOT NULL significa que la columna no puede contener ningún valor nulo. VARCHAR significa que la columna contendrá datos de caracteres de longitud variable. DATE significa que la columna contendrá información de fecha en un formato de tres partes (año, mes y día).

```
CREATE TABLE PROJ
  (PROJNO  CHAR(6)      NOT NULL,
   PROJNAME VARCHAR(24) NOT NULL,
   DEPTNO  CHAR(3)      NOT NULL,
   RESPEMP CHAR(6)      NOT NULL,
  PRSTAFF  DECIMAL(5,2) ,
  PRSTDATE DATE          ,
  PRENDATE DATE          ,
   MAJPROJ CHAR(6)      NOT NULL)
IN SCHED
```

Ejemplo 3: Cree una tabla llamada EMPLOYEE_SALARY donde los salarios desconocidos se consideren 0. No se especifica ningún espacio de tablas, de modo que la tabla se creará en un espacio de tablas seleccionado por el sistema basándose en las reglas descritas para la cláusula *IN nombre-espacio-tablas1*.

CREATE TABLE

```
CREATE TABLE EMPLOYEE_SALARY
(DEPTNO CHAR(3) NOT NULL,
DEPTNAME VARCHAR(36) NOT NULL,
EMPNO CHAR(6) NOT NULL,
SALARY DECIMAL(9,2) NOT NULL WITH DEFAULT)
```

Ejemplo 4: Cree tipos diferenciados para el total de salario y millas (kilómetros) y utilícelos para las columnas de una tabla creada en el espacio de tablas por omisión. En una sentencia de SQL dinámica, suponga que el registro especial CURRENT SCHEMA es JOHNDOE y el registro especial CURRENT PATH tiene el valor por omisión ("SYSIBM","SYSFUN","JOHNDOE").

Si no se especifica ningún valor para SALARY, debe establecerse en 0, y si no se especifica ningún valor para LIVING_DIST, debe establecerse en 1 milla (1,6 km).

```
CREATE DISTINCT TYPE JOHNDOE.T_SALARY AS INTEGER WITH COMPARISONS
```

```
CREATE DISTINCT TYPE JOHNDOE.MILES AS FLOAT WITH COMPARISONS
```

```
CREATE TABLE EMPLOYEE
(ID INTEGER NOT NULL,
NAME CHAR(30),
SALARY T_SALARY NOT NULL WITH DEFAULT,
LIVING_DIST MILES DEFAULT MILES(1) )
```

Ejemplo 5: Cree tipos diferenciados para la imagen y audio y utilícelos para las columnas de una tabla. No se especifica ningún espacio de tablas, de modo que la tabla se creará en un espacio de tablas seleccionado por el sistema basándose en las reglas descritas para la cláusula *IN nombre-espacio-tablas1*. Suponga que el registro especial CURRENT PATH tiene el valor por omisión.

```
CREATE DISTINCT TYPE IMAGE AS BLOB (10M)
```

```
CREATE DISTINCT TYPE AUDIO AS BLOB (1G)
```

```
CREATE TABLE PERSON
(SSN INTEGER NOT NULL,
NAME CHAR(30),
VOICE AUDIO,
PHOTO IMAGE)
```

Ejemplo 6: Cree la tabla EMPLOYEE en el espacio de tablas HUMRES. Las restricciones definidas en la tabla son las siguientes:

- Los valores del número de departamento deben estar comprendidos entre 10 y 100.
- El trabajo de un empleado sólo puede ser 'Sales', 'Mgr' o 'Clerk'.

- Aquellos empleados que lleven en la compañía desde 1986 deben ganar más de 40.500 dólares.

Nota: Si las columnas incluidas en las restricciones de comprobación pueden contener valores nulos, también podrían ser NULL.

```

CREATE TABLE EMPLOYEE
(ID          SMALLINT NOT NULL,
 NAME       VARCHAR(9),
 DEPT       SMALLINT CHECK (DEPT BETWEEN 10 AND 100),
 JOB        CHAR(5) CHECK (JOB IN ('Sales','Mgr','Clerk')),
 HIREDATE   DATE,
 SALARY     DECIMAL(7,2),
 COMM      DECIMAL(7,2),
 PRIMARY KEY (ID),
 CONSTRAINT YEARSAL CHECK (YEAR(HIREDATE) > 1986 OR SALARY > 40500)
)
IN HUMRES

```

Ejemplo 7: Cree una tabla que esté completamente contenida en el espacio de tablas PAYROLL.

```

CREATE TABLE EMPLOYEE .....
IN PAYROLL

```

Ejemplo 8: Cree una tabla con su parte de datos en ACCOUNTING y su parte de índice en ACCOUNT_IDX.

```

CREATE TABLE SALARY.....
IN ACCOUNTING INDEX IN ACCOUNT_IDX

```

Ejemplo 9: Cree una tabla y anote cronológicamente los cambios SQL en el formato por omisión.

```

CREATE TABLE SALARY1 .....

```

o

```

CREATE TABLE SALARY1 .....
DATA CAPTURE NONE

```

Ejemplo 10: Cree una tabla y anote cronológicamente los cambios SQL en un formato expandido.

```

CREATE TABLE SALARY2 .....
DATA CAPTURE CHANGES

```

Ejemplo 11: Cree una tabla EMP_ACT en el espacio de tablas SCHED. EMPNO, PROJNO, ACTNO, EMPTIME, EMSTDATE y EMENDATE son nombres de columna. Las restricciones definidas en la tabla son:

- El valor para el conjunto de columnas EMPNO, PROJNO y ACTNO en cualquier fila debe ser exclusivo.

CREATE TABLE

- El valor de PROJNO debe coincidir con un valor existente para la columna PROJNO de la tabla PROJECT y si se suprime el proyecto, también deben suprimirse todas las filas de EMP_ACT que hagan referencia al proyecto.

```
CREATE TABLE EMP_ACT
(EMPNO      CHAR(6) NOT NULL,
 PROJNO     CHAR(6) NOT NULL,
 ACTNO      SMALLINT NOT NULL,
 EMPTIME    DECIMAL(5,2),
  EMSTDATE  DATE,
 EMENDATE   DATE,
  CONSTRAINT EMP_ACT_UNIQ UNIQUE (EMPNO,PROJNO,ACTNO),
  CONSTRAINT FK_ACT_PROJ FOREIGN KEY (PROJNO)
REFERENCESPROJECT (PROJNO) ON DELETE CASCADE
)
IN SCHED
```

Se crea automáticamente un índice de unicidad denominado EMP_ACT_UNIQ en el mismo esquema para imponer la restricción de unicidad.

Ejemplo 12: Cree una tabla que vaya a contener información sobre goles famosos para el 'hall of fame' del hockey sobre hielo. La tabla listará información sobre el jugador que marcó el gol, el portero contra el que lo marcó, la fecha y el lugar, así como una descripción. Cuando se disponga de ello, también señalará los lugares en que se almacenan artículos de prensa sobre el partido e imágenes del gol fijas y en movimiento. Los artículos de prensa van a enlazarse, por lo que no pueden suprimirse ni renombrarse, pero deben seguir funcionando todas las aplicaciones de actualización y visualización existentes. Las imágenes fijas y en movimiento van a enlazarse con acceso bajo el control completo de DB2. Las imágenes fijas tendrán recuperación y se devolverán al propietario original si se desenlazan. Las imágenes en movimiento no tendrán recuperación y se suprimirán si se desenlazan. La columna de descripción y las tres columnas DATALINK tienen posibilidad de nulos.

```
CREATE TABLE HOCKEY_GOALS
( BY_PLAYER  VARCHAR(30) NOT NULL,
 BY_TEAM     VARCHAR(30) NOT NULL,
 AGAINST_PLAYER VARCHAR(30) NOT NULL,
 AGAINST_TEAM VARCHAR(30) NOT NULL,
 DATE_OF_GOAL DATE NOT NULL,
 DESCRIPTION CLOB(5000),
 ARTICLES    DATALINK LINKTYPE URL FILE LINK CONTROL MODE DB2OPTIONS,
 SNAPSHOT    DATALINK LINKTYPE URL FILE LINK CONTROL
              INTEGRITY ALL
              READ PERMISSION DB WRITE PERMISSION BLOCKED
              RECOVERY YES ON UNLINK RESTORE,
 MOVIE       DATALINK LINKTYPE URL FILE LINK CONTROL
              INTEGRITY ALL
              READ PERMISSION DB WRITE PERMISSION BLOCKED
              RECOVERY NO ON UNLINK DELETE )
```


Ejemplo 13: Supongamos que se necesita una tabla de excepción para la tabla EMPLOYEE. Se puede crear una utilizando la sentencia siguiente.

```
CREATE TABLE EXCEPTION_EMPLOYEE AS
(SELECT EMPLOYEE.*,
CURRENT_TIMESTAMP AS TIMESTAMP,
CAST (' ' AS CLOB(32K)) AS MSG
FROM EMPLOYEE
) DEFINITION ONLY
```

Ejemplo 14: Supongamos los espacios de tablas siguientes que tienen los atributos indicados:

TBSPACE	PAGESIZE	USER	USERAUTH
DEPT4K	4096	BOBBY	S
PUBLIC4K	4096	PUBLIC	S
DEPT8K	8192	BOBBY	S
DEPT8K	8192	RICK	S
PUBLIC8K	8192	PUBLIC	S

- Si RICK crea la tabla siguiente, se colocará en el espacio de tablas PUBLIC4K, pues el número de bytes es menor que 4005; pero si BOBBY crea la misma tabla, se colocará en el espacio de tablas DEPT4K, pues BOBBY tiene un privilegio USE otorgado explícitamente:

```
CREATE TABLE DOCUMENTS
(SUMMARY VARCHAR(1000),
REPORT VARCHAR(2000))
```

- Si BOBBY crea la tabla siguiente, se colocará en el espacio de tablas DEPT8K, pues el número de bytes es mayor que 4005, y BOBBY tiene un privilegio USE otorgado explícitamente. En cambio, si DUNCAN crea la misma tabla, se colocará en el espacio de tablas PUBLIC8K, pues DUNCAN no tiene ningún privilegio específico:

```
CREATE TABLE CURRICULUM
(SUMMARY VARCHAR(1000),
REPORT VARCHAR(2000),
EXERCISES VARCHAR(1500))
```

Ejemplo 15: Creación de una tabla que tiene una columna LEAD definida con el tipo estructurado EMP. Especifique 300 bytes para el valor INLINE LENGTH de la columna LEAD, lo cual significa que cualquier instancia de LEAD que no pueda caber dentro de los 300 bytes se almacenará fuera de la tabla (separadamente de la fila de la tabla base, de forma similar a como se manejan los valores LOB).

```
CREATE TABLE PROJECTS (PID INTEGER,
LEAD EMP INLINE LENGTH 300,
STARTDATE DATE,
...)
```

CREATE TABLE

Ejemplo 16: Creación de una tabla DEPT, con cinco columnas llamadas DEPTNO, DEPTNAME, MGRNO, ADMRDEPT y LOCATION. La columna DEPT debe definirse como columna de identidad para que DB2 genere siempre un valor para ella. Los valores de la columna DEPT deben comenzar en 500 y aumentar según incrementos de 1.

```
CREATE TABLE DEPT
  (DEPTNO SMALLINT NOT NULL
    GENERATED ALWAYS AS IDENTITY
    (START WITH 500, INCREMENT BY 1),
  DEPTNAME VARCHAR (36) NOT NULL,
MGRNO    CHAR(6),
  ADMRDEPT SMALLINT NOT NULL,
  LOCATION CHAR(30))
```

CREATE TABLESPACE

La sentencia CREATE TABLESPACE crea un nuevo espacio de tablas dentro de la base de datos, asigna contenedores al espacio de tablas y registra la definición y los atributos del espacio de tablas en el catálogo.

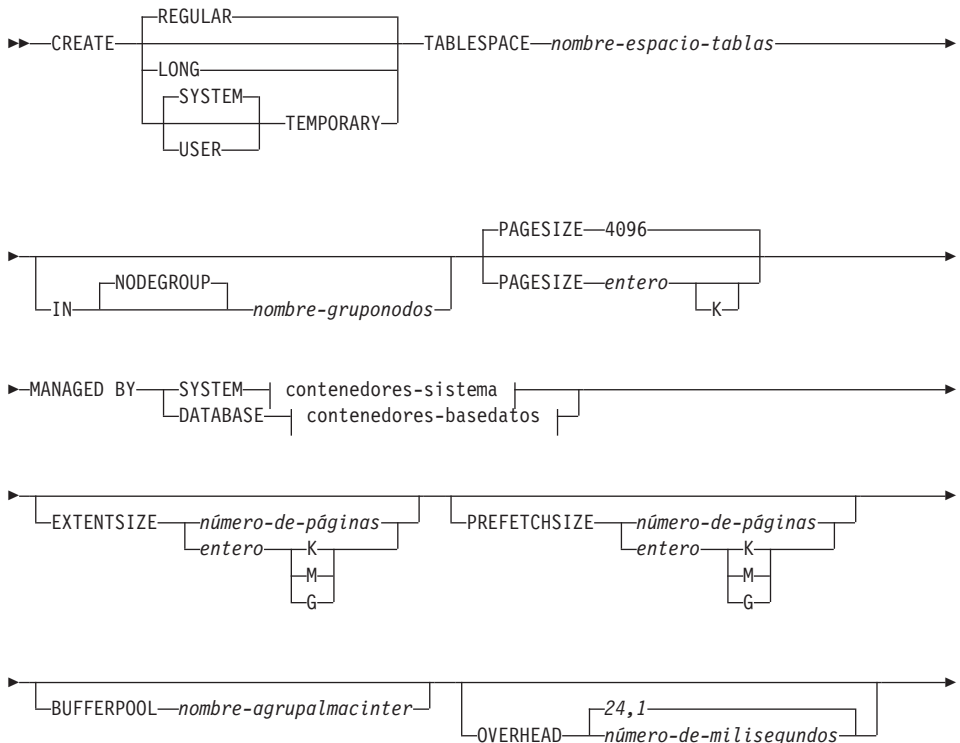
Invocación

Esta sentencia puede incluirse en un programa de aplicación o emitirse de forma interactiva. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

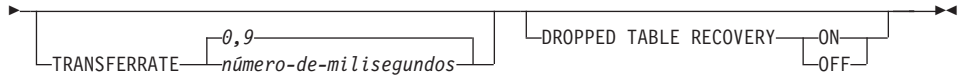
Autorización

El ID de autorización de la sentencia debe tener la autorización SYSCTRL o SYSADM.

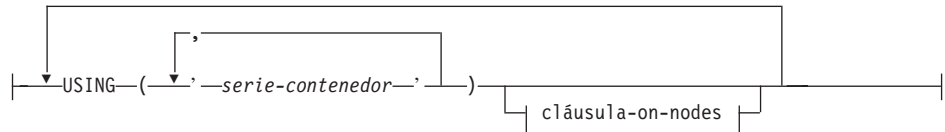
Sintaxis



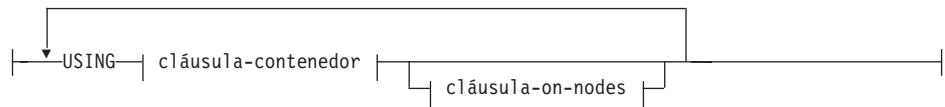
CREATE TABLESPACE



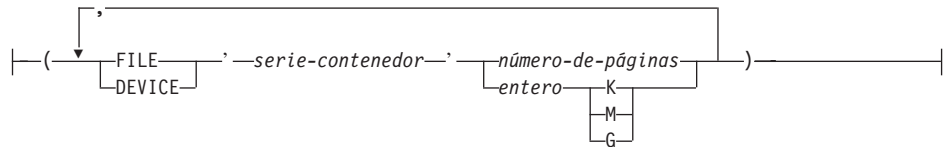
contenedores-sistema:



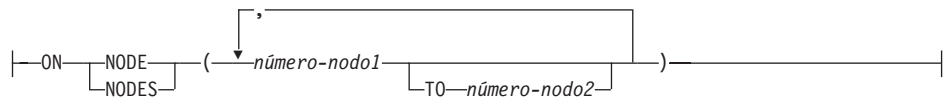
contenedores-basedatos:



cláusula-contenedor:



cláusula-on-nodes:



Descripción

REGULAR

Almacena todos los datos salvo las tablas temporales.

LONG

Almacena columnas de tabla largas o de tipo LOB. También puede contener columnas de tipo estructurado. El espacio de tablas debe ser un espacio de tablas DMS.

SYSTEM TEMPORARY

Almacena tablas temporales (son áreas de trabajo que el gestor de bases de datos utiliza para realizar operaciones tales como clasificaciones o uniones). La palabra clave SYSTEM es opcional. Observe que una base de datos debe tener siempre como mínimo un espacio de tablas SYSTEM TEMPORARY, pues las tablas temporales sólo pueden almacenarse en esa clase de espacio de tablas. Un espacio de tablas temporal se crea automáticamente cuando se crea una base de datos.

Vea CREATE DATABASE en el manual *Consulta de mandatos* para obtener más información.

USER TEMPORARY

Almacena tablas temporales globales declaradas. Observe que cuando se crea una base de datos no existe ningún espacio de tablas temporal de usuario. Debe crearse como mínimo un espacio de tablas temporal de usuario, con los privilegios USE apropiados, para poder definir tablas temporales declaradas.

nombre-espacio-tablas

Designa el espacio de tablas. Es un nombre que consta de una sola parte. Es un identificador de SQL (ordinario o delimitado). El *nombre-espacio-tablas* no debe identificar un espacio de tablas que ya exista en el catálogo (SQLSTATE 42710). El *nombre-espacio-tablas* no debe empezar por los caracteres SYS (SQLSTATE 42939).

IN NODEGROUP *nombre-gruponodos*

Especifica el grupo de nodos del espacio de tablas. El grupo de nodos debe existir. El único grupo de nodos que se puede especificar al crear un espacio de tablas SYSTEM TEMPORARY es IBMTEMPGROUP. La palabra clave NODEGROUP es opcional.

Si no se especifica el grupo de nodos, se utiliza el grupo de nodos por omisión (IBMDEFAULTGROUP) para los espacios de tablas REGULAR, LONG y USER TEMPORARY. Para los espacios de tablas SYSTEM TEMPORARY, se utiliza el grupo de nodos por omisión IBMTEMPGROUP.

PAGESIZE *entero* [K]

Define el tamaño de las páginas utilizadas para el espacio de tablas. Los valores válidos de *entero* sin el sufijo K son 4 096, 8 192, 16 384 ó 32 768. Los valores válidos de *entero* con el sufijo K son 4, 8, 16 ó 32. Se produce un error si el tamaño de página no es uno de estos valores (SQLSTATE 428DE) o no es igual al tamaño de página de la agrupación de almacenamientos intermedios del espacio de tablas (SQLSTATE 428CB). El valor por omisión es páginas de 4 096 bytes (4K). Se permite cualquier número de espacios entre *entero* y K, incluso ningún espacio.

CREATE TABLESPACE

MANAGED BY SYSTEM

Especifica que el espacio de tablas será un espacio gestionado por el sistema (SMS).

contenedores-sistema

Especifique los contenedores para un espacio de tablas SMS.

USING (*'serie-contenedor',...*)

En un espacio de tablas SMS, identifica uno o varios contenedores que pertenecerán al espacio de tablas y en los que se almacenarán los datos del espacio de tablas. La *serie-contenedor* no puede sobrepasar los 240 bytes de longitud.

Cada *serie-contenedor* puede ser un nombre de directorio absoluto o relativo. El nombre de directorio, si no es absoluto, será relativo al directorio de la base de datos. Si algún componente del nombre de directorio no existe, el gestor de bases de datos lo crea. Cuando se elimina un espacio de tablas, se suprimen todos los componentes creados por el gestor de bases de datos. Si existe el directorio identificado por *serie-contenedor*, éste no debe contener archivos ni subdirectorios (SQLSTATE 428B2).

El formato de *serie-contenedor* es dependiente del sistema operativo. El sistema operativo especifica los contenedores de la manera habitual. Por ejemplo, una vía de acceso de directorio OS/2 Windows 95 y Windows NT empieza por una letra de unidad seguida de ":", mientras que en los sistemas basados en UNIX, una vía de acceso empieza por "/".

Observe que no se da soporte a recursos remotos (por ejemplo, unidades redirigidas de LAN en OS/2, Windows 95 y Windows NT o sistemas de archivos montados NFS en AIX).

cláusula-on-nodes

Especifica la partición o las particiones en las que se crean los contenedores en una base de datos particionada. Si no se especifica esta cláusula, los contenedores se crean en las particiones del grupo de nodos que no están especificadas explícitamente en ninguna otra *cláusula-on-nodes*. Para un espacio de tablas SYSTEM TEMPORARY definido en el grupo de nodos IBMTEMPGROUP, cuando no se especifica la *cláusula-on-nodes*, los contenedores también se crearán en todas las nuevas particiones (nodos) añadidas a la base de datos. Vea la página 898 para obtener detalles sobre la especificación de esta cláusula.

MANAGED BY DATABASE

Especifica que el espacio de tablas será un espacio gestionado por la base de datos (espacio de tablas DMS).

contenedores-basedatos

Especifique los contenedores para un espacio de tablas DMS.

USING

Introduce una cláusula-contenedor.

cláusula-contenedor

Especifica los contenedores para un espacio de tablas DMS.

(FILE | DEVICE 'serie-contenedor' número-de-páginas,...)

Para un espacio de tablas DMS, identifica uno o más contenedores que pertenecerán al espacio de tablas y donde se almacenarán los datos del espacio de tablas. Se especifican el tipo del contenedor (FILE o DEVICE) y su tamaño (en páginas de PAGESIZE). El tamaño también puede especificarse como un valor entero seguido de K (para kilobytes), M (para megabytes) o G (para gigabytes). Si se especifica de esta manera, el nivel mínimo del número de bytes dividido por el tamaño de página se utiliza para determinar el número de páginas para el contenedor. Se puede especificar una mezcla de contenedores FILE y DEVICE. La *serie-contenedor* no puede sobrepasar los 254 bytes de longitud.

En un contenedor FILE, la *serie-contenedor* debe ser un nombre de archivo relativo o absoluto. El nombre de archivo, si no es absoluto, será relativo al directorio de la base de datos. Si algún componente del nombre de directorio no existe, el gestor de bases de datos lo crea. Si el archivo no existe, se creará e inicializará con el tamaño especificado por el gestor de bases de datos. Cuando se elimina un espacio de tablas, se suprimen todos los componentes creados por el gestor de bases de datos.

Nota: Si el archivo existe, se sobregabarará y si es menor a lo especificado, se ampliará. El archivo no se truncará si es más grande de lo que se ha especificado.

Para un contenedor DEVICE, la *serie-contenedor* debe ser un nombre de dispositivo. El dispositivo ya debe existir.

Todos los contenedores deberán ser exclusivos en todas las bases de datos; un contenedor solamente puede pertenecer a un espacio de tablas. El tamaño de los contenedores puede ser diferente, aunque el rendimiento ideal se consigue cuando todos los contenedores tienen el mismo tamaño. El formato exacto *serie-contenedor* depende del sistema operativo. El sistema operativo especificará los contenedores de la manera habitual. Para obtener más detalles sobre la declaración de contenedores, consulte el manual Administration Guide.

CREATE TABLESPACE

No se da soporte a los recursos remotos (por ejemplo unidades redirigidas de LAN en OS/2, Windows 95 y Windows NT o sistemas de archivos montados NFS en AIX).

cláusula-on-nodes

Especifica la partición o particiones en las que se crean las particiones en una base de datos particionada. Si no se especifica esta cláusula, los contenedores se crean en las particiones del grupo de nodos que no están especificadas explícitamente en ninguna otra *cláusula-on-nodes*. Para un espacio de tablas SYSTEM TEMPORARY definido en el grupo de nodos IBMTEMPGROUP, cuando no se especifica la *cláusula-on-nodes*, los contenedores también se crearán en todas las nuevas particiones añadidas a la base de datos. Vea la página 898 para obtener detalles sobre la especificación de esta cláusula.

cláusula-on-nodes

Especifica las particiones en las que se crean los contenedores en una base de datos particionada.

ON NODES

Palabras clave que indican que se han especificado particiones específicas. NODE es sinónimo de NODES.

número-nodo1

Especifique un número de partición (o nodo) específico.

TO *número-nodo2*

Especifique un rango de números de particiones (o nodos). El valor de *número-nodo2* debe ser mayor o igual que el valor de *número-nodo1* (SQLSTATE 428A9). Todas las particiones comprendidas en el rango especificado de números de partición son las particiones para las que se crean los contenedores si el nodo pertenece al grupo de nodos del espacio de tablas.

La partición especificada por el número y cada partición (o nodo) en el rango de particiones debe existir en el grupo de nodos en el que está definido el espacio de tablas (SQLSTATE 42729). Un número de partición sólo puede aparecer explícitamente o en un rango en una *cláusula-on-nodes* exactamente para la sentencia (SQLSTATE 42613).

EXTENTSIZE *número-de-páginas*

Especifica el número de páginas de PAGESIZE que se grabarán en un contenedor antes de pasar al siguiente contenedor. El valor de EXTENTSIZE también puede especificarse como un valor entero seguido de K (para kilobytes), M (para megabytes) o G (para gigabytes). Si se especifica de esta manera, el nivel mínimo del número de bytes dividido por el tamaño de página se utiliza para

determinar el valor del número de páginas para EXTENTSIZE. El gestor de bases de datos pasa periódicamente por los contenedores a medida que se almacenan datos.

El parámetro de configuración DFT_EXTENT_SZ proporciona el valor por omisión.

PREFETCHSIZE *número-de-páginas*

Especifica el número de páginas de PAGESIZE que se leerán del espacio de tablas cuando se realice la lectura anticipada de datos. El valor del tamaño de lectura anticipada también puede especificarse como un valor entero seguido de K (para kilobytes), M (para megabytes) o G (para gigabytes). Si se especifica de esta manera, el nivel mínimo del número de bytes dividido por el tamaño de página se utiliza para determinar el valor del número de páginas para el tamaño de lectura anticipada. La lectura anticipada lee los datos que una consulta necesita antes que la consulta haga referencia a ellos, por lo que la consulta no necesita esperar a que se efectúen operaciones de E/S.

El parámetro de configuración DFT_PREFETCH_SZ proporciona el valor por omisión. (Este parámetro de configuración, como todos los parámetros de configuración, se explica con detalle en el manual *Administration Guide*.)

BUFFERPOOL *nombre-agrupalmacinter*

El nombre de la agrupación de almacenamientos intermedios utilizado para las tablas de este espacio de tablas. La agrupación de almacenamientos intermedios debe existir (SQLSTATE 42704). Si no se especifica, se utiliza la agrupación de almacenamientos intermedios por omisión (IBMDEFAULTBP). El tamaño de página de la agrupación de almacenamientos intermedios debe coincidir con el tamaño de página especificado (o el valor por omisión) para el espacio de tablas (SQLSTATE 428CB). El grupo de nodos del espacio de tablas debe estar definido para la agrupación de almacenamientos intermedios (SQLSTATE 42735).

OVERHEAD *número-de-milisegundos*

Cualquier literal numérico (entero, decimal o de coma flotante) que especifique la actividad general del controlador de E/S y el tiempo de búsqueda y latencia del disco, en milisegundos. Para todos los contenedores que pertenecen al espacio de tablas, este número debe ser un promedio (o, en todo caso, el mismo número). Este valor sirve para determinar el coste de E/S durante la optimización de una consulta.

TRANSFERRATE *número-de-milisegundos*

Cualquier literal numérico (entero, decimal o de coma flotante) que especifique el tiempo para leer una página en la memoria, en

CREATE TABLESPACE

milisegundos. Para todos los contenedores que pertenecen al espacio de tablas, este número debe ser un promedio (o, en todo caso, el mismo número). Este valor sirve para determinar el coste de E/S durante la optimización de una consulta.

DROPPED TABLE RECOVERY

Las tablas eliminadas del espacio de tablas especificado pueden recuperarse mediante la opción RECOVER TABLE ON del mandato ROLLFORWARD. Esta cláusula sólo puede especificarse para un espacio de tablas REGULAR (SQLSTATE 42613). Para obtener más información sobre la recuperación de tablas eliminadas, vea el manual *Administration Guide*.

Notas

- Para obtener información sobre cómo determinar los valores correctos de EXTENTSIZE, PREFETCHSIZE, OVERHEAD y TRANSFERRATE, consulte el manual *Administration Guide*.
- Elegir entre un espacio gestionado por la base de datos o un espacio gestionado por el sistema para un espacio de tablas es una elección importante en la que intervienen ventajas e inconvenientes. Consulte el manual *Administration Guide* para ver una exposición de estas ventajas e inconvenientes.
- Cuando en una base de datos hay más de un espacio de tablas TEMPORARY, se utilizarán en modalidad rotatoria para equilibrar su utilización. Consulte el manual *Administration Guide* para obtener información sobre la utilización de más de un espacio de tablas, el equilibrio y los valores recomendados para EXTENTSIZE, PREFETCHSIZE, OVERHEAD y TRANSFERRATE.
- En una base de datos particionadas si más de una partición reside en el mismo nodo físico, no se puede especificar el mismo dispositivo o vía de acceso específica para dichas particiones (SQLSTATE 42730). Para este entorno, especifique una *serie-contenedor* exclusiva para cada partición o utilice un nombre de vía de acceso relativo.
- Puede especificar una expresión de nodo para la sintaxis de serie-contenedor cuando cree contenedores de SMS o DMS. Normalmente, se especifica la expresión de nodo si se utilizan diversos nodos lógicos del sistema de bases de datos particionadas. Esto asegura que los nombres de los contenedores sean exclusivos para los nodos (servidores de particiones de bases de datos). Cuando especifica la expresión, el número de nodo forma parte del nombre de contenedor o, si especifica argumentos adicionales, forma parte de dicho nombre el resultado del argumento. Se utiliza el argumento “ \$N” ([blanco]\$N) para indicar la expresión de nodo. El argumento debe aparecer al final de la serie-contenedor y sólo puede utilizarse con uno de los siguientes formatos. En la tabla que aparece a continuación, se supone que el número de nodo es 5:

Tabla 26. Argumentos para la creación de contenedores

Sintaxis	Ejemplo	Valor
[blanco]\$N	" \$N"	5
[blanco]\$N+[número]	" \$N+1011"	1016
[blanco]\$N%[número]	" \$N%3"	2
[blanco]\$N+[número]%[número]	" \$N+12%13"	4
[blanco]\$N%[número]+[número]	" \$N%3+20"	22
Nota: - % es un módulo - En todos los casos, los operadores se evalúan de izquierda a derecha.		

A continuación, se encuentran algunos ejemplos:

Ejemplo 1:

```
CREATE TABLESPACE TS1 MANAGED BY DATABASE USING  
(device '/dev/rcont $N' 20000)
```

En un sistema de dos nodos, se utilizarían los contenedores siguientes:

```
/dev/rcont0 - on NODE 0  
/dev/rcont1 - on NODE 1
```

Ejemplo 2:

```
CREATE TABLESPACE TS2 MANAGED BY DATABASE USING  
(file '/DB2/containers/TS2/container $N+100' 10000)
```

En un sistema de cuatro nodos, se crearían los contenedores siguientes:

```
/DB2/containers/TS2/container100 - on NODE 0  
/DB2/containers/TS2/container101 - on NODE 1  
/DB2/containers/TS2/container102 - on NODE 2  
/DB2/containers/TS2/container103 - on NODE 3
```

Ejemplo 3:

```
CREATE TABLESPACE TS3 MANAGED BY SYSTEM USING  
( '/TS3/cont $N%2', '/TS3/cont $N%2+2')
```

En un sistema de dos nodos, se crearían los contenedores siguientes:

```
/TS3/cont0 - On NODE 0  
/TS3/cont2 - On NODE 0  
/TS3/cont1 - On NODE 1  
/TS3/cont3 - On NODE 1
```

CREATE TABLESPACE

Ejemplos

Ejemplo 1: Cree un espacio de tablas DMS normal en un sistema basado en UNIX que utiliza 3 dispositivos de 10 000 páginas de 4K cada una. Especifique sus características de E/S.

```
CREATE TABLESPACE PAYROLL  
MANAGED BY DATABASE  
USING (DEVICE '/dev/rhdisk6' 10000,  
         DEVICE '/dev/rhdisk7' 10000,  
         DEVICE '/dev/rhdisk8' 10000)  
OVERHEAD 24.1  
TRANSFERRATE 0.9
```

Ejemplo 2: Cree un espacio de tablas SMS normal en OS/2 o en Windows NT que utiliza 3 directorios en tres unidades separadas, con un tamaño de extensión de 64 páginas y un tamaño de lectura anticipada de 32 páginas.

```
CREATE TABLESPACE ACCOUNTING  
MANAGED BY SYSTEM  
USING ('d:\acc_tbsp', 'e:\acc_tbsp', 'f:\acc_tbsp')  
EXTENTSIZE 64  
PREFETCHSIZE 32
```

Ejemplo 3: Cree un espacio de tablas DMS temporal en Unix que utiliza 2 archivos de 50.000 páginas cada uno y un tamaño de extensión de 256 páginas.

```
CREATE TEMPORARY TABLESPACE TEMPSPACE2  
MANAGED BY DATABASE  
USING (FILE '/tmp/tempSPACE2.f1' 50000,  
         FILE '/tmp/tempSPACE2.f2' 50000)  
EXTENTSIZE 256
```

Ejemplo 4: Cree un espacio de tablas DMS en el grupo de nodos ODDNODEGROUP (nodos 1,3,5) en una base de datos particionada Unix. En todas las particiones (o nodos), utilice el dispositivo /dev/rhdisk0 para 10 000 páginas de 4K. Especifique también un dispositivo específico de partición para cada partición con 40.000 páginas de 4K.

```
CREATE TABLESPACE PLANS  
MANAGED BY DATABASE  
USING (DEVICE '/dev/rhdisk0' 10000, DEVICE '/dev/rn1hd01' 40000)  
ON NODE (1)  
USING (DEVICE '/dev/rhdisk0' 10000, DEVICE '/dev/rn3hd03' 40000)  
ON NODE (3)  
USING (DEVICE '/dev/rhdisk0' 10000, DEVICE '/dev/rn5hd05' 40000)  
ON NODE (5)
```

CREATE TRANSFORM

La sentencia CREATE TRANSFORM define funciones de transformación, identificadas por un nombre de grupo, que se utilizan para intercambiar valores de tipo estructurado con programas del lenguaje principal y con funciones y métodos externos.

Invocación

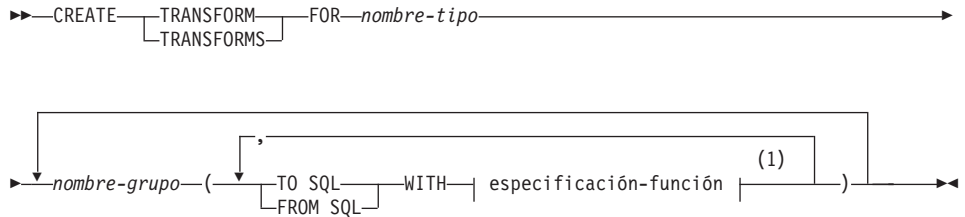
Esta sentencia se puede intercalar en un programa de aplicación o se puede emitir utilizando sentencias de SQL dinámicas. Es una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

Autorización

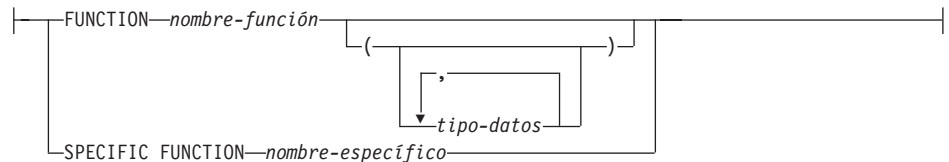
El ID de autorización de la sentencia debe tener al menos uno de los siguientes privilegios:

- Autorización SYSADM o DBADM
- Definidor del tipo identificado por *nombre-tipo* y definidor de cada función especificada.

Sintaxis



especificación-función:



Notas:

- 1 Una misma cláusula no se debe especificar más de una vez.

CREATE TRANSFORM

Descripción

TRANSFORM o TRANSFORMS

Indica que se están definiendo uno o más grupos de transformación. Se puede especificar cualquiera de las dos versiones de la palabra clave.

FOR *nombre-tipo*

Especifica un nombre para el tipo estructurado definido por el usuario para el cual se define el grupo de transformación.

En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un *nombre-tipo* no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para un *nombre-tipo* no calificado. El *nombre-tipo* debe ser el nombre de un tipo existente definido por el usuario (SQLSTATE 42704) y debe ser un tipo estructurado (SQLSTATE 42809). El tipo estructurado o cualquier otro tipo estructurado incluido en la misma jerarquía de tipos no debe tener transformaciones ya definidas con el nombre-grupo proporcionado (SQLSTATE 42739).

nombre-grupo

Especifica el nombre del grupo de transformación donde residen las funciones TO SQL y FROM SQL. No es necesario que el nombre sea exclusivo, pero si un grupo de transformación tiene este nombre (con la misma instrucción TO SQL y/o FROM SQL definida), no debe estar definido previamente para el *nombre-tipo* especificado (SQLSTATE 42739). Un *nombre-grupo* debe ser un identificador SQL, con una longitud máxima de 18 caracteres (SQLSTATE 42622) y no puede contener ningún prefijo calificador (SQLSTATE 42601). El *nombre-grupo* no puede comenzar con el prefijo 'SYS', pues éste está reservado para su uso por la base de datos (SQLSTATE 42939).

Como máximo, se puede especificar una de las clases siguientes de funciones FROM SQL y TO SQL para un grupo dado cualquiera (SQLSTATE 42628).

TO SQL

Define la función específica que se utiliza para transformar un valor al formato del tipo estructurado SQL definido por el usuario. La función debe tener todos sus parámetros como tipos de datos incorporados y el tipo devuelto es *nombre-tipo*.

FROM SQL

Define la función específica que se utiliza para transformar un valor a un tipo de datos incorporado representativo del tipo estructurado SQL definido por el usuario. La función debe tener un parámetro del tipo de datos *nombre-tipo* y devolver un tipo de datos incorporado (o conjunto de tipos de datos incorporados).

WITH *especificación-función*

Hay varias maneras de especificar la instancia de función.

Si se especifica FROM SQL, la *especificación-función* debe identificar una función que cumpla los requisitos siguientes:

- existe un parámetro del tipo *nombre-tipo*
- el tipo devuelto es un tipo incorporado o una fila cuyas columnas todas tienen tipos incorporados
- la signatura especifica LANGUAGE SQL o la utilización de otra transformación FROM SQL que está definida con LANGUAGE SQL.

Si se especifica TO SQL, la *especificación-función* debe identificar una función que cumpla los requisitos siguientes:

- todos los parámetros tienen tipos incorporados
- el tipo devuelto es *nombre-tipo*
- la signatura especifica LANGUAGE SQL o la utilización de otra transformación TO SQL que está definida con LANGUAGE SQL.

Los métodos (aunque se especifiquen con FUNCTION ACCESS) no se pueden especificar como transformaciones mediante *especificación-función*. En su lugar, sólo las funciones que están definidas por la sentencia CREATE FUNCTION puede actuar como transformaciones (SQLSTATE 42704 ó 42883).

Además, aunque no es obligatorio, los tipos incorporados devueltos por la función FROM SQL se deben corresponder directamente con los tipos incorporados que son parámetros de la función TO SQL. Esto es una consecuencia lógica de la relación inversa que existe entre estas dos funciones.

FUNCTION *nombre-función*

Identifica la función específica por su nombre, y sólo es válido si hay exactamente una función con el *nombre-función*. La función identificada puede tener un número cualquiera de parámetros definidos para ella.

En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objetos no calificados.

Si no existe ninguna función con este nombre en el esquema nombrado o implícito, se produce un error (SQLSTATE 42704). Si hay más de una instancia específica de la función en el esquema nombrado o implícito, se produce un error (SQLSTATE 42725). El algoritmo estándar de selección de funciones no se utiliza.

CREATE TRANSFORM

FUNCTION *nombre-función (tipo-datos,...)*

Proporciona la signatura de la función, que identifica de manera exclusiva la función que se debe utilizar. El algoritmo estándar de selección de funciones no se utiliza.

nombre-función

Especifica el nombre de la función. En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objetos no calificados.

(tipo-datos,...)

Los tipos de datos especificados aquí deben coincidir con los tipos de datos que se han especificado en la posición correspondiente de la sentencia CREATE FUNCTION. El número de tipos de datos y la concatenación lógica de los tipos de datos se utilizan para identificar la función específica.

Si el tipo-datos no está calificado, el nombre de tipo se resuelve efectuando una búsqueda en los esquemas de la vía de acceso de SQL. Esto también se aplica a los nombres de tipo de datos especificados para un tipo REFERENCE.

Para los tipos de datos parametrizados no es necesario especificar la longitud, la precisión ni la escala. En su lugar, se puede codificar un conjunto vacío de paréntesis para indicar que esos atributos deben eludirse al comparar tipos de datos.

No puede utilizarse FLOAT() (SQLSTATE 42601), pues el valor del parámetro indica tipos de datos diferentes (REAL o DOUBLE). De todas formas, si la longitud, la precisión o la escala están codificadas, el valor debe coincidir exactamente con el que se especifica en la sentencia CREATE FUNCTION.

No es necesario que un tipo de FLOAT(n) coincida con el valor definido para n puesto que $0 < n < 25$ significa REAL y $24 < n < 54$ significa DOUBLE. La coincidencia se produce basándose en si el tipo es REAL o DOUBLE. Observe que el atributo FOR BIT DATA no se considera parte de la signatura por lo que respecta a la comparación de patrones. Por ejemplo, un CHAR FOR BIT DATA especificado en la signatura coincidiría con una función definida con CHAR solamente.

Si en el esquema nombrado o implícito no existe ninguna función con la signatura especificada, se produce un error (SQLSTATE 42883).

SPECIFIC FUNCTION *nombre-especifico*

Identifica la función definida por el usuario, mediante un nombre que se especifica o se toma por omisión al crear la función.

En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objeto no calificados. El *nombre-especifico* debe identificar una instancia de función específica del esquema nombrado o implícito; de lo contrario se produce un error (SQLSTATE 42704).

Notas

- Cuando no se especifica un grupo de transformación en un programa de aplicación (mediante la opción de precompilación/enlace TRANSFORM GROUP para el SQL estático, o utilizando la sentencia SET CURRENT DEFAULT TRANSFORM GROUP para el SQL dinámico), las funciones de transformación del grupo de transformación 'DB2_PROGRAM' se utilizan (si están definidas) cuando el programa de aplicación recupera o envía variables del lenguaje principal que están basadas en el tipo estructurado definido por el usuario identificado por *nombre-tipo*. Cuando se recupera un valor del tipo de datos *nombre-tipo*, se ejecuta la transformación FROM SQL para transformar el tipo estructurado al tipo de datos incorporado devuelto por la función de transformación. Similarmente, cuando se envía una variable del lenguaje principal que se asignará a un valor del tipo de datos *nombre-tipo*, se ejecuta la transformación TO SQL para transformar el valor del tipo de datos incorporado al valor del tipo estructurado. Si no se especifica un grupo de transformación definido por el usuario o no se define un grupo 'DB2_PROGRAM' (para el tipo estructurado proporcionado), se produce un error.
- El tipo de datos incorporado correspondiente a una variable de lenguaje principal de tipo estructurado debe ser asignable:
 - desde el resultado de la función de transformación FROM SQL para el tipo estructurado tal como está definida por la opción especificada TRANSFORM GROUP del mandato de precompilación (utilizando reglas de asignación para la recuperación) y
 - al parámetro de la función de transformación TO SQL para el tipo estructurado tal como está definida por la opción especificada TRANSFORM GROUP del mandato de precompilación (utilizando reglas de asignación de memoria).

Si una variable de lenguaje principal no es compatible con el tipo necesario para la correspondiente función de transformación, se produce un error (para el enlace de entrada: SQLSTATE 42821, para el enlace de salida:

CREATE TRANSFORM

SQLSTATE 42806). Para conocer los errores que resultan de la asignación de series de caracteres, vea “Asignaciones de series” en la página 107.

- Las funciones de transformación identificadas en el grupo de transformación por omisión llamado 'DB2_FUNCTION' se utilizan siempre que se invoca una función definida por el usuario, no escrita en SQL, utilizando el tipo de datos *nombre-tipo* como parámetro o tipo devuelto. Esto es aplicable cuando la función o método no especifica la cláusula TRANSFORM GROUP. Cuando se invoca la función con argumento del tipo de datos *nombre-tipo*, se ejecuta la transformación FROM SQL para transformar el tipo estructurado al tipo de datos incorporado devuelto por la función de transformación. Similarmente, cuando el tipo de datos devuelto de la función es *nombre-tipo*, se ejecuta la transformación TO SQL para transformar el valor del tipo de datos incorporado, devuelto por la función externa, al valor del tipo estructurado.
- Si un tipo estructurado contiene un atributo que es también un tipo estructurado, las funciones de transformación asociadas deben expandir (o ensamblar) recursivamente todos los tipos estructurados anidados. Esto significa que los resultados o parámetros de las funciones de transformación constan solamente del conjunto de tipos incorporados representativos de todos los atributos base del tipo estructurado sujeto (incluidos todos sus tipos estructurados anidados). Las funciones de transformación no se propagan en cascada para procesar tipos estructurados anidados.
- La función o funciones identificadas en esta sentencia se resuelven durante la ejecución de la sentencia, de acuerdo con las reglas descritas anteriormente. Cuando estas funciones se utilizan (implícitamente) en sentencias de SQL subsiguientes, no son objeto de otro proceso de resolución. Las funciones de transformación definidas en esta sentencia se registran exactamente tal como se resuelven en esta sentencia.
- Cuando se crean o eliminan atributos o subtipos de un tipo determinado, también deben modificarse las funciones de transformación correspondientes al tipo estructurado definido por el usuario.
- Para un grupo de transformación determinado, las funciones FROM SQL y TO SQL se pueden especificar en la misma cláusula *nombre-grupo*, en cláusulas *nombre-grupo* separadas o en sentencias CREATE TRANSFORM separadas. La única restricción es que una función determinada FROM SQL o TO SQL no se puede volver a definir sin antes eliminar la definición de grupo existente. Esto le permite, por ejemplo, definir primero una función de transformación FROM SQL para un grupo determinado y más tarde definir la correspondiente función de transformación TO SQL para el mismo grupo.

Ejemplos

Ejemplo 1: Creación de dos grupos de transformación que asocian el tipo estructurado POLYGON definido por el usuario con una función de transformación personalizada para C y una especializada para Java.

```
CREATE TRANSFORM FOR POLYGON
  mystruct1 (FROM SQL WITH FUNCTION myxform_sqlstruct,
             TO SQL WITH FUNCTION myxform_structsql)
  myjava1   (FROM SQL WITH FUNCTION myxform_sqljava,
             TO SQL WITH FUNCTION myxform_javasql )
```

CREATE TRIGGER

CREATE TRIGGER

La sentencia CREATE TRIGGER define un desencadenante en una base de datos.

Invocación

Esta sentencia se puede intercalar en un programa de aplicación o se puede emitir utilizando sentencias de SQL dinámicas. Es una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar dinámicamente (SQLSTATE 42509).

Autorización

Cuando se crea el desencadenante, el ID de autorización de la sentencia debe tener como mínimo uno de los privilegios siguientes:

- Autorización SYSADM o DBADM.
- Privilegio ALTER para la tabla donde está definido el desencadenante, o el privilegio ALTERIN para el esquema de la tabla donde está definido el desencadenante y uno de los elementos siguientes:
 - Autorización IMPLICIT_SCHEMA para la base de datos, si no existe el nombre de esquema implícito ni explícito del desencadenante.
 - Privilegio CREATEIN para el esquema, si el nombre de esquema del desencadenante hace referencia a un esquema existente.

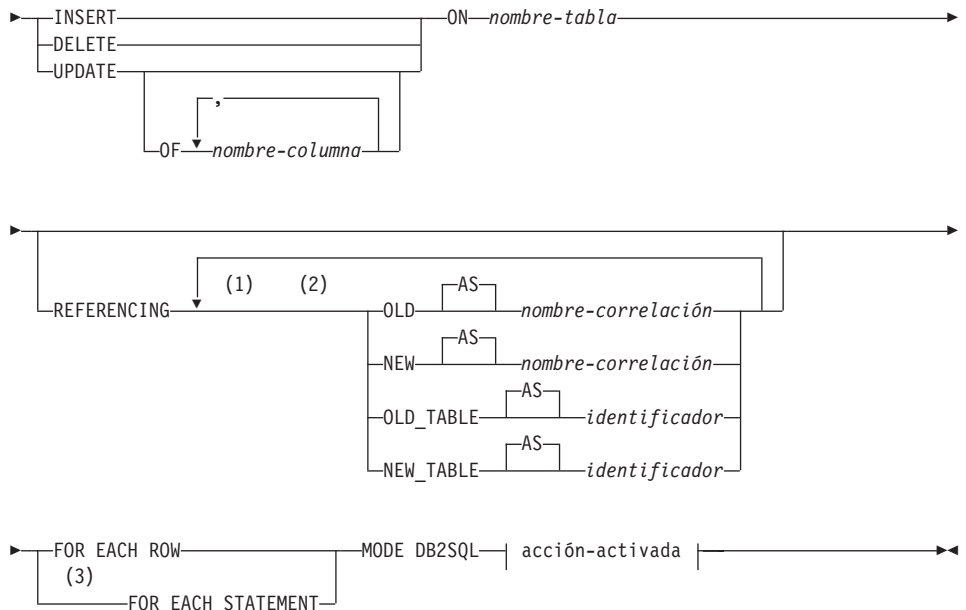
Si el ID de autorización de la sentencia no tiene la autorización SYSADM ni DBADM, los privilegios que tiene el ID de autorización de la sentencia (sin tener en cuenta los privilegios PUBLIC ni de grupo) deben incluir todos los siguientes elementos, siempre que exista el desencadenante:

- Privilegio SELECT para la tabla donde está definido el desencadenante, si se especifica cualquier tabla o variable de transición.
- Privilegio SELECT para cualquier tabla o vista referenciada en la condición de la acción activada.
- Los privilegios necesarios para invocar las sentencias de SQL activadas especificadas.

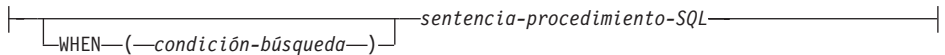
Si la persona que define un desencadenante sólo puede crear el desencadenante porque tiene autorización SYSADM, entonces se le otorga la autorización DBAM explícita para crear el desencadenante.

Sintaxis

```
►► CREATE TRIGGER nombre-desencadenante [NO CASCADE BEFORE | AFTER] ►
```



acción-activada:



Notas:

- 1 Puede especificarse OLD y NEW sólo una vez cada uno.
- 2 Puede especificarse OLD_TABLE y NEW_TABLE sólo una vez cada uno para los desencadenantes AFTER.
- 3 Es posible no especificar FOR EACH STATEMENT para los desencadenantes BEFORE.

Descripción

nombre-desencadenante

Indica el nombre del desencadenante. El nombre, incluido el nombre de esquema implícito o explícito, no debe designar un desencadenante que ya esté descrito en el catálogo (SQLSTATE 42710). Si se especifica un nombre compuesto de dos partes, el nombre de esquema no puede empezar por "SYS" (SQLSTATE 42939).

NO CASCADE BEFORE

Especifica que la acción activada asociada se debe aplicar antes de aplicar a la base de datos los cambios ocasionados por la actualización real de la

CREATE TRIGGER

tabla sujeto. También especifica que la acción activada del desencadenante no provocará la activación de otros desencadenantes.

AFTER

Especifica que la acción activada asociada se debe aplicar después de que los cambios ocasionados por la actualización real y la tabla sujeto se apliquen a la base de datos.

INSERT

Especifica que la acción activada asociada con el desencadenante se ejecutará siempre que se aplique una operación INSERT en la tabla base designada.

DELETE

Especifica que la acción activada asociada con el desencadenante se ejecutará siempre que una operación DELETE se aplique a la tabla base designada.

UPDATE

Especifica que la acción activada asociada con el desencadenante se ejecutará siempre que una operación UPDATE se aplique a la tabla sujeto base designada en las columnas especificadas o implicadas.

Si no se especifica la lista *nombre-columna* opcional, están implicadas todas las columnas de la tabla. Por lo tanto, la omisión de la lista *nombre-columna* supone la activación del desencadenante cuando se actualiza cualquier columna de la tabla.

OF *nombre-columna*,...

Cada *nombre-columna* especificada debe ser una columna de la tabla base (SQLSTATE 42703). Si el desencadenante es un desencadenante anterior, el *nombre-columna* especificado no puede ser una columna generada distinta de la columna de identidad (SQLSTATE 42989). Ningún *nombre-columna* deberá aparecer más de una vez en la lista *nombre-columna* (SQLSTATE 42711). El desencadenante sólo se podrá activar al actualizar una columna que conste en la lista *nombre-columna*.

ON *nombre-tabla*

Designa la tabla sujeto de la definición del desencadenante. El nombre debe especificar una tabla base o un seudónimo que dé como resultado una tabla base (SQLSTATE 42809). El nombre no debe especificar una tabla de catálogo (SQLSTATE 42832), una tabla de resumen (SQLSTATE 42997), una tabla temporal declarada (SQLSTATE 42995) ni un apodo (SQLSTATE 42809).

REFERENCING

Especifica los nombres de correlación para las *variables de transición* y los nombres de tabla para las *tablas de transición*. Los nombres de correlación identifican una fila determinada del conjunto de filas que se ven afectadas

por la operación SQL desencadenante. Los nombres de tabla identifican todo el conjunto de filas afectadas. Cada fila afectada por la operación SQL desencadenante está disponible para la acción activada mediante la calificación de las columnas con *nombres-correlación* especificados de la siguiente manera:

OLD AS *nombre-correlación*

Especifica un nombre de correlación que identifica el estado de fila anterior a la operación SQL desencadenante.

NEW AS *nombre-correlación*

Especifica un nombre de correlación que identifica el estado de la fila tal como la ha modificado la operación SQL desencadenante y cualquier sentencia SET de un desencadenante BEFORE que ya se ha ejecutado.

La acción activada dispone del conjunto completo de filas afectadas por la operación SQL desencadenante mediante la utilización de un nombre de tabla temporal que se especifica de la siguiente manera:

OLD_TABLE AS *identificador*

Especifica un nombre de tabla temporal que identifica el conjunto de filas afectadas antes de la operación SQL desencadenante.

NEW_TABLE AS *identificador*

Especifica un nombre de tabla temporal que identifica las filas afectadas tal como las ha modificado la operación SQL desencadenante y cualquier sentencia SET de un desencadenante BEFORE que ya se ha ejecutado.

Las siguientes normas se aplican a la cláusula REFERENCING:

- Ninguno de los nombres de correlación OLD y NEW y los nombres de OLD_TABLE y NEW_TABLE pueden ser idénticos (SQLSTATE 42712).
- Para un desencadenante solamente se puede especificar un *nombre-correlación* OLD y uno NEW (SQLSTATE 42613).
- Para un desencadenante solamente se puede especificar un *identificador* OLD_TABLE y uno NEW_TABLE (SQLSTATE 42613).
- El *nombre-correlación* OLD y el *identificador* OLD_TABLE solamente se pueden utilizar si el suceso desencadenante es una operación DELETE o UPDATE (SQLSTATE 42898). Si la operación es DELETE, el *nombre-correlación* OLD captura el valor de la fila suprimida. Si la operación es UPDATE, captura el valor de la fila antes de la operación UPDATE. Ocurre exactamente lo mismo con el *identificador* OLD_TABLE y el conjunto de filas afectadas.
- El *nombre-correlación* NEW y el *identificador* NEW_TABLE solamente se pueden utilizar si el suceso desencadenante es una operación INSERT o UPDATE (SQLSTATE 42898). En ambas operaciones, el valor de NEW

CREATE TRIGGER

captura el nuevo estado de la fila como lo proporciona la operación original y modificado por cualquier desencadenante BEFORE que se haya ejecutado hasta ese momento. Ocurre exactamente lo mismo con el *identificador* NEW_TABLE y el conjunto de filas afectadas.

- Los *identificadores* OLD_TABLE y NEW_TABLE no se pueden definir para un desencadenante BEFORE (SQLSTATE 42898).
- Los *nombres-correlación* no se pueden definir para un desencadenante FOR EACH STATEMENT (SQLSTATE 42899).
- Las tablas de transición no pueden modificarse (SQLSTATE 42807).
- El total de las referencias a las columnas de la tabla de transición y a las variables de transición de la acción activada no puede sobrepasar el límite del número de columnas de una tabla o la suma de sus longitudes no puede exceder la longitud máxima de una fila de una tabla (SQLSTATE 54040).
- El ámbito de cada *nombre-correlación* y de cada *identificador* es la totalidad de la definición del desencadenante.

FOR EACH ROW

Especifica que la acción activada se debe aplicar una vez en cada fila de la tabla sujeto que se haya visto afectada por la operación SQL desencadenante.

FOR EACH STATEMENT

Especifica que la acción activada se debe aplicar una vez para toda la sentencia. Este tipo de granularidad de desencadenante no se puede especificar para un desencadenante BEFORE (SQLSTATE 42613). Si se especifica, se activa una sentencia desencadenante UPDATE o DELETE incluso cuando no existen filas afectadas por la sentencia UPDATE o DELETE desencadenante.

MODE DB2SQL

Esta cláusula sirve para especificar la modalidad de los desencadenantes. En este momento es la única modalidad válida a la que se da soporte.

acción-activada

Especifica la acción que se debe realizar cuando se activa un desencadenante. Una acción activada se compone de una *sentencia-procedimiento-SQL* y de una condición opcional para la ejecución de la *sentencia-procedimiento-SQL*.

WHEN (*condición-búsqueda*)

Especifica una condición verdadera, falsa o desconocida. La *condición-búsqueda* proporciona la posibilidad de determinar si se debe ejecutar o no una determinada acción activada.

La acción asociada se realiza sólo si la condición de búsqueda especificada es verdadera. Si se omite la cláusula WHEN, la *sentencia-procedimiento-SQL* asociada se ejecuta siempre.

sentencia-procedimiento-SQL

La *sentencia-procedimiento-SQL* puede contener una sentencia dinámica compuesta o cualquiera de las sentencias de control de SQL listadas en el apartado “SQL compuesto (dinámico)” en la página 642.

Si el desencadenante es un desencadenante BEFORE, una *sentencia-procedimiento-SQL* también puede incluir uno de los elementos siguientes:

- una selección completa⁸⁵
- una sentencia SET variable.

Si el desencadenante es un desencadenante AFTER, una *sentencia-procedimiento-SQL* puede incluir también uno de los elementos siguientes:

- una sentencia INSERT de SQL
- una sentencia UPDATE de SQL
- una sentencia DELETE de SQL
- una sentencia SET variable
- una selección completa⁸⁵

La *sentencia-procedimiento-SQL* no debe contener una sentencia no soportada (SQLSTATE 42987).

La *sentencia-procedimiento-SQL* no puede hacer referencia a una variable de transición no definida (SQLSTATE 42703) ni a una tabla temporal declarada (SQLSTATE 42995).

La *sentencia-procedimiento-SQL* en un desencadenante BEFORE no puede hacer referencia a una tabla de resumen definida con REFRESH IMMEDIATE (SQLSTATE 42997).

La *sentencia-procedimiento-SQL* de un desencadenante BEFORE no puede hacer referencia a una columna generada, distinta de la columna de identidad, en la nueva variable de transición (SQLSTATE 42989).

Notas

- Al añadir un desencadenante a una tabla que ya contiene filas, no provocará la activación de ninguna acción activada. Así pues, si el desencadenante tiene como objetivo imponer las restricciones de la tabla, es posible que las filas existentes no cumplan dichas restricciones.

85. Una expresión-común-tabla debe preceder a una selección completa.

CREATE TRIGGER

- Si los sucesos para dos desencadenantes tienen lugar simultáneamente (por ejemplo, si tienen el mismo suceso, tiempo de activación y tablas sujeto), el primer desencadenante creado es el primero en ejecutarse.
- Si se añade una columna a la tabla sujeto cuando ya se han definido los desencadenantes, se aplican las siguientes reglas:
 - Si se trata de un desencadenante UPDATE que se ha especificado sin una lista de columnas explícita, cualquier actualización de una columna nueva provocará la activación del desencadenante.
 - La columna no estará visible en la acción activada de cualquier desencadenante definido con anterioridad.
 - Las tablas de transición OLD_TABLE y NEW_TABLE no incluirán esta columna. Por este motivo, el resultado de "SELECT *" en una tabla de transición no contendrá la columna añadida.
- Si se añade una columna a cualquier tabla a la que se haga referencia en una acción activada, la nueva columna no estará visible para la acción activada.
- El resultado de una selección completa especificada en la *sentencia-procedimiento-SQL* no está disponible dentro ni fuera del desencadenante.
- Un desencadenante BEFORE DELETE definido en una tabla implicada en un ciclo de restricciones de referencia en cascada, no debería incluir referencias a la tabla en la que está definido ni a ninguna otra tabla modificada en cascada durante la evaluación del ciclo de restricciones de integridad de referencia. El resultado de tal desencadenante son datos dependientes y, por lo tanto, tal vez no produzcan resultados coherentes. En su forma más simple, esto significa que un desencadenante BEFORE DELETE en una tabla con una restricción de referencia para sí misma y la norma de supresión CASCADE, no debería incluir ninguna referencia a la tabla en la *acción-activada*.
- La creación de un desencadenante hace que se marquen determinados paquetes como no válidos:
 - Si se crea un desencadenante de actualización sin una lista de columnas explícita, se invalidan los paquetes con una utilización de actualización sobre la tabla de destino.
 - Si se crea un desencadenante de actualización con una lista de columnas, sólo se invalidan los paquetes con utilización de actualización en la tabla de destino si el paquete también tiene una utilización de actualización como mínimo en una columna de la lista *nombre-columna* de la sentencia CREATE TRIGGER.
 - Si se crea un desencadenante de inserción, se invalidan en la tabla de destino los paquetes que tienen una utilización de inserción.
 - Si se crea un desencadenante de supresión, se invalidan en la tabla de destino los paquetes que tienen una utilización de supresión.

- Un paquete permanece invalidado hasta que el programa de aplicación se enlaza explícitamente, se vuelve a enlazar o se ejecuta y el gestor de bases de datos lo vuelve a enlazar de manera automática.
- **Desencadenantes no operativos:** Un *desencadenante no operativo* es un desencadenante que ya no está disponible y que, por consiguiente, no se activa nunca. Un desencadenante deja de ser operativo si:
 - se revoca un privilegio que el creador del desencadenante debe tener para que se ejecute el desencadenante
 - se elimina un objeto, por ejemplo una tabla, una vista o un seudónimo, del que depende la acción activada
 - deja de estar operativa una vista de la que depende la acción activada
 - se elimina un seudónimo que es la tabla sujeto del desencadenante.

En otras palabras, un desencadenante no operativo es aquél en el que se ha eliminado la definición de un desencadenante a consecuencia de las reglas en cascada de las sentencias DROP y REVOKE. Por ejemplo, cuando se elimina una vista, deja de estar operativo cualquier desencadenante con una *sentencia-procedimiento-SQL* definida utilizando dicha vista.

Cuando un desencadenante deja de ser operativo, todos los paquetes con sentencias que realicen operaciones y que estuvieran activando el desencadenante quedarán marcados como no válidos. Cuando el paquete se vuelve a enlazar (explícita o implícitamente), se ignora por completo el desencadenante no operativo. De igual forma, las aplicaciones con sentencias de SQL dinámicas que realicen operaciones y que estuvieran activando el desencadenante también pasarán por alto por completo todos los desencadenantes que no sean operativos.

El nombre del desencadenante puede seguirse especificando en las sentencias DROP TRIGGER y COMMENT ON TRIGGER.

Es posible volver a crear un desencadenante no operativo emitiendo una sentencia CREATE TRIGGER y utilizando el texto de definición del desencadenante no operativo. Este texto de definición de desencadenante se almacena en la columna TEXT de la vista de catálogo SYSCAT.TRIGGERS. Observe que no es necesario eliminar de manera explícita el desencadenante no operativo para volver a crearlo. La emisión de una sentencia CREATE TRIGGER con el mismo *nombre-desencadenante* que un desencadenante no operativo hará que se sustituya dicho desencadenante no operativo con un aviso (SQLSTATE 01595).

Los desencadenantes no operativos se señalan con una X en la columna VALID de la vista de catálogo SYSCAT.TRIGGERS.

CREATE TRIGGER

- **Errores al ejecutar desencadenantes:** Los errores que se producen durante la ejecución de sentencias de SQL activadas se devuelven utilizando SQLSTATE 09000 a no ser que el error se considere grave. Si el error es grave, se devuelve el SQLSTATE de error grave. El campo SQLERRMC de la SQLCA de un error no grave incluirá el nombre de desencadenante, el SQLCODE, el SQLSTATE y tantos símbolos de la anomalía como quepan. La *sentencia-procedimiento-SQL* puede incluir una sentencia SIGNAL SQLSTATE o una función RAISE_ERROR. En ambos casos, el SQLSTATE que se devuelve es el que se especifica en la sentencia SIGNAL SQLSTATE o la condición RAISE_ERROR.
- La creación de un desencadenante con un nombre de esquema que todavía no existe dará como resultado la creación implícita de dicho esquema siempre que el ID de autorización de la sentencia tenga la autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN para el esquema se otorga a PUBLIC.
- Antes de ejecutar cualquier desencadenante BEFORE, el gestor de bases de datos crea un valor para una columna de identidad. Por lo tanto, el desencadenante BEFORE puede acceder al valor de identidad generado.
- Después de ejecutar todos desencadenante BEFORE, el gestor de bases de datos crea un valor para una columna generada por expresión. Por lo tanto, los desencadenantes BEFORE no pueden acceder al valor generado por la expresión.
- **Desencadenantes y tablas con tipo:** Se puede asociar un desencadenante a una tabla con tipo para un nivel cualquiera de una jerarquía de tablas. Si una sentencia de SQL activa varios desencadenantes, éstos se ejecutarán en el orden en que fueron creados, aunque estén asociados a tablas diferentes de la jerarquía de tablas con tipo.

Cuando un desencadenante está activado, sus variables de transición (OLD, NEW, OLD_TABLE y NEW_TABLE) pueden contener filas de subtablas. Sin embargo, sólo contendrán columnas definidas en la tabla a la que están asociadas.

Efectos de las sentencias INSERT, UPDATE y DELETE:

- Desencadenantes de fila: Cuando se utiliza una sentencia de SQL para insertar, actualizar o suprimir una fila de tabla, la sentencia activa desencadenantes de fila asociados a la tabla más específica donde reside la fila, y los desencadenantes asociados a todas las supertablas de esa tabla. Esta regla se cumple siempre, cualquiera que sea la forma en que la sentencia de SQL accede a la tabla. Por ejemplo, al emitir un mandato UPDATE EMP, algunas de las filas actualizadas pueden estar en la subtabla MGR. Para las filas de EMP, se activan los desencadenantes de fila asociados a EMP y a sus supertablas. Para las filas de MGR, se activan los desencadenantes de fila asociados a MGR y a sus supertablas.
- Desencadenantes de sentencia: Las sentencias INSERT, UPDATE o DELETE activan desencadenantes de sentencia asociados a las tablas (y a

sus supertablas) que podrían estar afectados por la sentencia. Esta regla se cumple siempre, con independencia de si hay realmente filas afectadas por la sentencia en esas tablas. Por ejemplo, en un mandato INSERT INTO EMP, se activan los desencadenantes de sentencia para EMP y sus supertablas. Otro ejemplo: En un mandato UPDATE EMP o DELETE EMP, se activan los desencadenantes para EMP y sus supertablas y subtablas, aunque no se haya actualizado ni suprimido ninguna fila de subtabla. Del mismo modo, un mandato UPDATE ONLY (EMP) o DELETE ONLY (EMP) activará desencadenantes de sentencia para EMP y sus supertablas, pero no desencadenantes de sentencia para subtablas.

Efectos de las sentencias DROP TABLE: Una sentencia DROP TABLE ("eliminar tabla") no activa ningún desencadenante asociado a la tabla que se elimina. En cambio, si la tabla eliminada es una subtabla, todas las filas de la tabla eliminada son elegibles para ser suprimidas de sus supertablas. Por lo tanto, para una tabla T:

- Desencadenantes de fila: DROP TABLE T activa desencadenantes de supresión de filas que están asociados a todas las supertablas de T, para cada fila de T.
- Desencadenantes de sentencia: DROP TABLE T activa desencadenantes de supresión de sentencias que están asociados a todas las supertablas de T, sin importar si T contiene o no alguna fila.

Acciones sobre vistas: Para predecir qué desencadenantes son activados por una acción sobre una vista, utilice la definición de la vista para convertir esa acción en una acción sobre tablas base. Por ejemplo:

1. Una sentencia de SQL ejecuta UPDATE V1, donde V1 es una vista con tipo y V2 es una subvista de ella. Suponga que V1 deriva de la tabla T1, y V2 deriva de la tabla T2. Esta sentencia podría potencialmente afectar a filas de T1, T2 y de sus subtablas, por lo tanto se activan desencadenantes de sentencia para T1 y T2 y para todas sus subtablas y supertablas.
2. Una sentencia de SQL ejecuta UPDATE V1, donde V1 es una vista con tipo y V2 es una subvista de ella. Suponga que V1 está definida como SELECT ... FROM ONLY(T1) y V2 está definida como SELECT ... FROM ONLY(T2). Debido a que la sentencia no puede afectar a filas de subtablas de T1 y T2, se activan desencadenantes de sentencia para T1 y T2 y para sus supertablas, pero no para sus subtablas.
3. Una sentencia de SQL ejecuta UPDATE ONLY(V1), donde V1 es una vista con tipo que está definida como SELECT ... FROM T1. La sentencia puede potencialmente afectar a T1 y a sus subtablas. Por lo tanto, se activan desencadenantes de sentencia para T1 y para todas sus subtablas y supertablas.

CREATE TRIGGER

- Una sentencia de SQL ejecuta UPDATE ONLY(V1), donde V1 es una vista con tipo que está definida como SELECT ... FROM ONLY(T1). En este caso, T1 es la única tabla que puede verse afectada por la sentencia, aunque V1 tenga subvistas y T1 tenga subtablas. Por lo tanto, se activan desencadenantes de sentencia sólo para T1 y sus supertablas.

Ejemplos

Ejemplo 1: Cree dos desencadenantes que den como resultado un seguimiento automático del número de empleados que gestiona una empresa. Los desencadenantes interactuarán con las tablas siguientes:

La tabla EMPLOYEE con estas columnas: ID, NAME, ADDRESS y POSITION.

La tabla COMPANY_STATS con estas columnas: NBEMP, NBPRODUCT y REVENUE.

El primer desencadenante aumenta el número de empleados cada vez que se da de alta a una persona; es decir, cada vez que se inserta una nueva fila en la tabla EMPLOYEE.

```
CREATE TRIGGER NEW_HIRED
AFTER INSERT ON EMPLOYEE
FOR EACH ROW MODE DB2SQL
UPDATE COMPANY_STATS SET NBEMP = NBEMP + 1
```

El segundo desencadenante reduce el número de empleados cada vez que un empleado se marcha de la compañía; es decir, cada vez que se suprime una fila de la tabla EMPLOYEE:

```
CREATE TRIGGER FORMER_EMP
AFTER DELETE ON EMPLOYEE
FOR EACH ROW MODE DB2SQL
UPDATE COMPANY_STATS SET NBEMP = NBEMP - 1
```

Ejemplo 2: Cree un desencadenante que asegure que siempre que se actualice un registro de piezas, se lleven a cabo la siguiente comprobación y acción (si es necesaria):

Si la cantidad disponible es inferior al 10% de la cantidad máxima de existencias, se emitirá una petición de entrega solicitando el número de artículos de la pieza en cuestión sea la cantidad máxima en existencias menos la cantidad disponible.

El desencadenante interactuará con la tabla PARTS con estas columnas: PARTNO, DESCRIPTION, ON_HAND, MAX_STOCKED y PRICE.

ISSUE_SHIP_REQUEST es una función definida por el usuario que envía una hoja de pedido de piezas adicionales a la compañía adecuada.

```
CREATE TRIGGER REORDER
AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
REFERENCING NEW AS N
```

```

FOR EACH ROW MODE DB2SQL
WHEN (N.ON_HAND < 0.10 * N.MAX_STOCKED)
BEGIN ATOMIC
VALUES(ISSUE_SHIP_REQUEST(N.MAX_STOCKED - N.ON_HAND, N.PARTNO));
END

```

Ejemplo 3: Cree un desencadenante que emita un error cuando se produzca una actualización que daría como resultado un aumento de salario mayor que el diez por ciento del salario actual.

```

CREATE TRIGGER RAISE_LIMIT
AFTER UPDATE OF SALARY ON EMPLOYEE
REFERENCING NEW AS N OLD AS O
FOR EACH ROW MODE DB2SQL
WHEN (N.SALARY > 1.1 * O.SALARY)
    SIGNAL SQLSTATE '75000' ('Aumento salarial>10%')

```

Ejemplo 4: Suponga que una aplicación registra y hace un seguimiento de los cambios en los precios de las existencias. Esta base de datos contiene dos tablas CURRENTQUOTE y QUOTEHISTORY.

Tablas: CURRENTQUOTE (SYMBOL, QUOTE, STATUS)
 QUOTEHISTORY (SYMBOL, QUOTE, QUOTE_TIMESTAMP)

Al actualizar la columna QUOTE de CURRENTQUOTE, el nuevo precio debería copiarse, con una indicación horaria, en la tabla QUOTEHISTORY. Asimismo, la columna STATUS de CURRENTQUOTE debería actualizarse para reflejar si las existencias:

1. son un valor en alza;
2. están en valor máximo del año;
3. son un valor a la baja;
4. están en el valor mínimo del año;
5. son un valor estable.

Las sentencias CREATE TRIGGER que realizan este cometido son las siguientes:

- Definición del desencadenante para determinar el estado:

```

CREATE TRIGGER STOCK_STATUS
NO CASCADE BEFORE UPDATE OF QUOTE ON CURRENTQUOTE
REFERENCING NEW AS NEWQUOTE OLD AS OLDQUOTE
FOR EACH ROW MODE DB2SQL
BEGIN ATOMIC
SET NEWQUOTE.STATUS =
CASE
WHEN NEWQUOTE.QUOTE >
    (SELECT MAX(QUOTE) FROM QUOTEHISTORY
    WHERE SYMBOL = NEWQUOTE.SYMBOL
    AND YEAR(QUOTE_TIMESTAMP) = YEAR(CURRENT DATE) )
THEN 'Alto'
WHEN NEWQUOTE.QUOTE <
    (SELECT MIN(QUOTE) FROM QUOTEHISTORY

```

CREATE TRIGGER

```
        WHERE SYMBOL = NEWQUOTE.SYMBOL
        AND YEAR(QUOTE_TIMESTAMP) = YEAR(CURRENT DATE) )
    THEN 'Bajo'
WHEN NEWQUOTE.QUOTE > OLDQUOTE.QUOTE
    THEN 'En alza'
WHEN NEWQUOTE.QUOTE < OLDQUOTE.QUOTE
    THEN 'A la baja'
WHEN NEWQUOTE.QUOTE = OLDQUOTE.QUOTE
    THEN 'Estable'
END;
END
```

- Definición del desencadenante para registrar un cambio en la tabla QUOTEHISTORY:

```
CREATE TRIGGER RECORD_HISTORY
AFTER UPDATE OF QUOTE ON CURRENTQUOTE
REFERENCING NEW AS NEWQUOTE
FOR EACH ROW MODE DB2SQL
BEGIN ATOMIC
    INSERT INTO QUOTEHISTORY
        VALUES (NEWQUOTE.SYMBOL, NEWQUOTE.QUOTE, CURRENT_TIMESTAMP);
END
```


CREATE TYPE (Estructurado)

La sentencia CREATE TYPE define un tipo estructurado definido por el usuario. Un tipo estructurado definido por el usuario puede incluir cero o más atributos. Un tipo estructurado puede ser un subtipo que permita que los atributos se hereden de un supertipo. La ejecución satisfactoria de la sentencia genera métodos para recuperar y actualizar valores de atributos. La ejecución satisfactoria de la sentencia también genera funciones para crear instancias de un tipo estructurado usado en una columna, convertir entre el tipo de referencia y su tipo de representación, y para dar soporte a los operadores de comparación (=, <>, <, <=, > y >=) para el tipo de referencia.

La sentencia CREATE TYPE también define especificaciones de método para métodos definidos por el usuario, para su uso con el tipo estructurado definido por el usuario.

Invocación

Esta sentencia se puede intercalar en un programa de aplicación o se puede emitir utilizando sentencias de SQL dinámicas. Es una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

Autorización

El ID de autorización de la sentencia debe tener como mínimo uno de los privilegios siguientes:

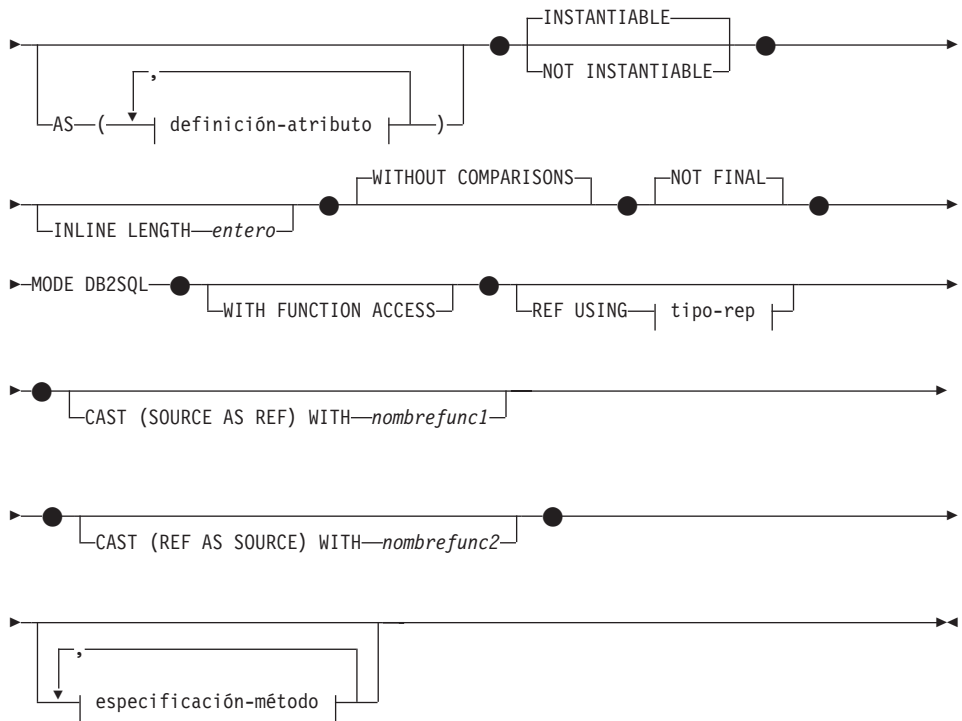
- Autorización SYSADM o DBADM
- Autorización IMPLICIT_SCHEMA para la base de datos si el nombre de esquema del tipo no hace referencia a un esquema existente.
- Privilegio CREATEIN para el esquema si el nombre de esquema del tipo hace referencia a un esquema existente.

Si se especifica UNDER y el ID de autorización de la sentencia no es el mismo que el definidor del tipo raíz de la jerarquía de tipos, entonces es necesaria la autorización SYSADM o DBADM.

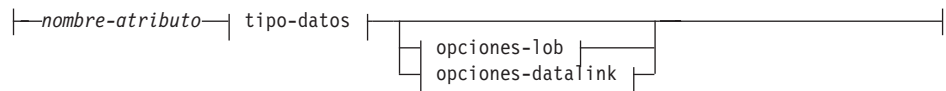
Sintaxis

```
►► CREATE TYPE nombre-tipo ───────────────────────────────────────────────────────────►  
└── UNDER nombre-supertipo ─────────────────────────────────────────────────────────┘
```

CREATE TYPE (Estructurado)

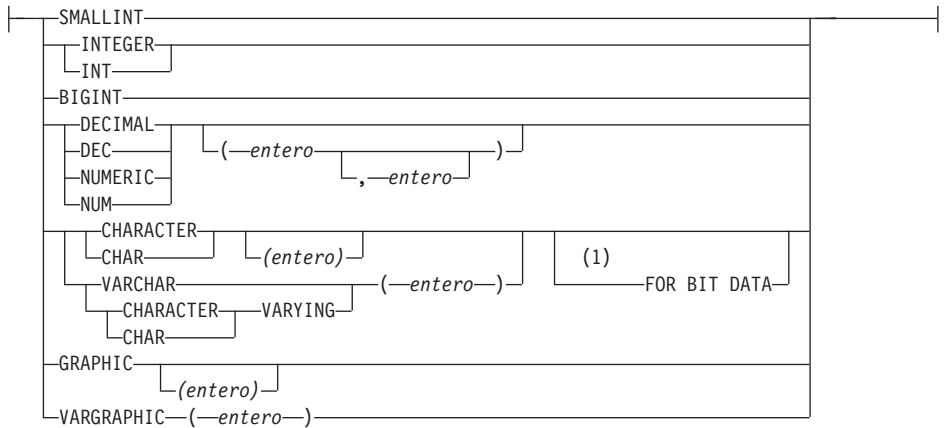


definición-atributo:

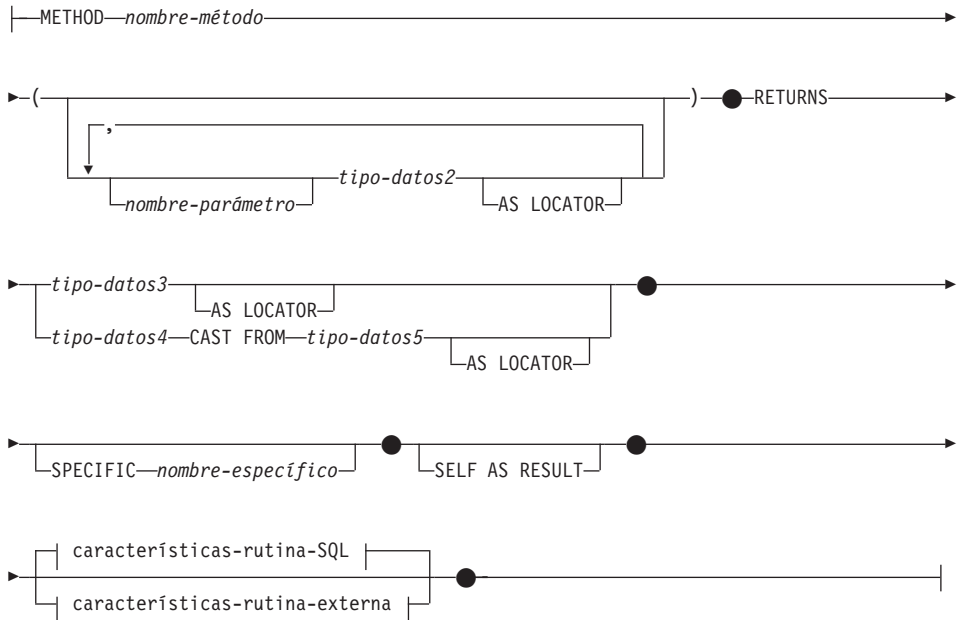


tipo-rep:

CREATE TYPE (Estructurado)



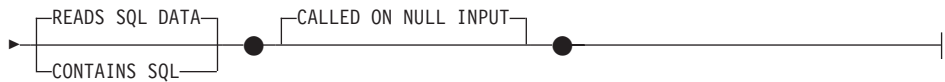
especificación-método:



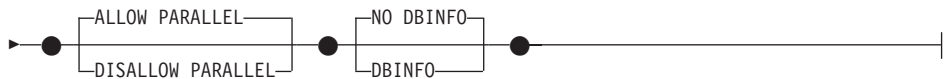
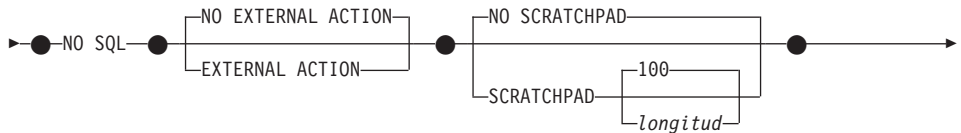
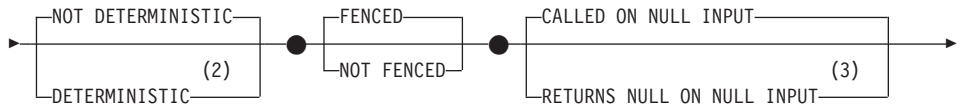
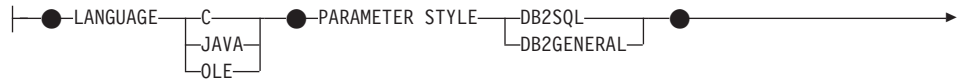
características-rutina-SQL:



CREATE TYPE (Estructurado)



características-rutina-externa:



Notas:

- 1 La cláusula FOR BIT DATA puede especificarse en orden aleatorio con las restricciones de columna siguientes.
- 2 Puede especificarse NOT VARIANT en lugar de DETERMINISTIC y VARIANT puede especificarse en lugar de NOT DETERMINISTIC.
- 3 Puede especificarse NULL CALL en lugar de CALLED ON NULL INPUT y puede especificarse NOT NULL CALL en lugar de RETURNS NULL ON NULL INPUT.

Descripción

nombre-tipo

Indica el tipo. El nombre, incluido el calificador implícito o explícito, no debe identificar ningún otro tipo (incorporado, estructurado o

diferenciado) ya descrito en el catálogo. El nombre no calificado no debe ser el mismo que el de un tipo de datos incorporado o BOOLEAN (SQLSTATE 42918). En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objetos no calificados.

El nombre de esquema (implícito o explícito) no debe ser mayor que 8 bytes (SQLSTATE 42622).

Algunos nombres utilizados como palabras clave en predicados están reservados para su uso por el sistema, y no pueden utilizarse como *nombre-tipo* (SQLSTATE 42939). Estos nombres son SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH y los operadores de comparación que se describen en el apartado “Predicado básico” en la página 213.

Si se especifica un *nombre-tipo* de dos partes, el nombre de esquema no puede empezar por “SYS”; de lo contrario, se genera un error (SQLSTATE 42939).

UNDER *nombre-supertipo*

Especifica que este tipo estructurado es un subtipo por debajo del *nombre-supertipo* especificado. El *nombre-supertipo* debe identificar un tipo estructurado existente (SQLSTATE 42704). Si *nombre-supertipo* está especificado sin un nombre de esquema, el tipo se resuelve realizando una búsqueda en los esquemas de la vía de acceso de SQL. El tipo estructurado incluye todos los atributos del supertipo seguidos de los atributos adicionales que se proporcionan en la *definición-atributo*.

definición-atributo

Define los atributos del tipo estructurado.

nombre-atributo

El nombre de un atributo. El *nombre-atributo* no puede ser el mismo que el de otro atributo de este tipo estructurado o un supertipo de este tipo estructurado (SQLSTATE 42711).

Algunos nombres utilizados como palabras clave en predicados están reservados para su uso por el sistema, y no pueden utilizarse como *nombre-atributo* (SQLSTATE 42939). Estos nombres son SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH y los operadores de comparación que se describen en el apartado “Predicado básico” en la página 213.

tipo-datos

El tipo de datos del atributo. Es uno de los tipos de datos listados en el apartado “CREATE TABLE” en la página 835, diferente de LONG

CREATE TYPE (Estructurado)

VARCHAR, LONG VARCHAR y cualquier tipo diferenciado basado en LONG VARCHAR o LONG VARCHAR (SQLSTATE 42601). El tipo de datos debe identificar un tipo de datos existente (SQLSTATE 42704). Si *tipo-datos* está especificado sin un nombre de esquema, el tipo se resuelve realizando una búsqueda en los esquemas de la vía de acceso de SQL. La descripción de diversos tipos de datos se proporciona en el apartado “CREATE TABLE” en la página 835. Si el tipo de datos del atributo es un tipo de referencia, el tipo destino de la referencia debe ser un tipo estructurado existente o se crea mediante esta sentencia (SQLSTATE 42704).

Si un tipo estructurado está definido con un atributo de tipo DATALINK, sólo puede utilizarse de manera efectiva como tipo de datos para una tabla con vista o vista con tipo (SQLSTATE 01641).

Para evitar que las definiciones de tipo que, durante la ejecución, permitirían que una instancia del tipo contenga, directa o indirectamente, otra instancia del mismo tipo o uno de sus subtipos, no se puede definir un tipo de forma que uno de sus tipos de atributo haga uso de sí mismo directa o indirectamente (SQLSTATE 428EP). Vea “Tipos estructurados” en la página 97 para obtener más información.

opciones-lob

Especifica las opciones asociadas con tipos LOB (o tipos diferenciados basados en tipos LOB). Para obtener una descripción detallada de *opciones-lob*, consulte el apartado “CREATE TABLE” en la página 835.

opciones-datalink

Especifica las opciones asociadas con tipos DATALINK (o tipos diferenciados basados en tipos DATALINK). Para obtener una descripción detallada de *opciones-datalink*, consulte el apartado “CREATE TABLE” en la página 835.

Tenga en cuenta que, si no se especifican opciones para un tipo DATALINK o un tipo diferenciado cuya fuente sea DATALINK, los valores por omisión serán las opciones LINKTYPE URL y NO LINK CONTROL.

INSTANTIABLE o NOT INSTANTIABLE

Determina si se puede crear una instancia del tipo estructurado. El no poder crear una instancia de un tipo estructurado tiene estas consecuencias:

- no se genera ninguna función constructora para un tipo para el que no se pueden crear instancias
- el tipo que no admite creación de instancias no puede utilizarse como tipo de una tabla o vista (SQLSTATE 428DP)

- el tipo que no admite creación de instancias se puede utilizar como tipo de una columna (sólo pueden insertarse en la columna valores nulos o instancias de subtipos que permiten la creación de instancias.

Para crear instancias de un tipo que no permite crear instancias, se deben crear subtipos que permiten crear instancias. Si se especifica NOT INSTANTIABLE, no se puede crear ninguna instancia del nuevo tipo.

INLINE LENGTH *entero*

Esta opción indica el tamaño máximo, en bytes, de una instancia de columna de tipo estructurado para su almacenamiento "inline" con el resto de valores de la fila de una tabla. Las instancias de un tipo estructurado o de sus subtipos que son mayores que la longitud "inline" especificada se almacenan separadamente de la fila de la tabla base, de forma similar a como se manejan los valores LOB.

Si el valor especificado de INLINE LENGTH es menor que el tamaño del resultado de la función constructora para el tipo recién creado (32 bytes más 10 bytes por atributo) y menor que 292 bytes, se produce un error (SQLSTATE 429B2). Observe que el número de atributos incluye todos los atributos heredados a partir del supertipo del tipo.

El valor INLINE LENGTH del tipo, ya sea especificado explícitamente o tomado por omisión, es la longitud "inline" por omisión de las columnas que hacen uso del tipo estructurado. Este valor por omisión se puede sustituir por otro valor al crear la tabla.

El valor INLINE LENGTH no es aplicable cuando el tipo estructurado se utiliza como tipo de una tabla con tipo.

El valor por omisión de INLINE LENGTH para un tipo estructurado es calculado por el sistema. En la fórmula mostrada más abajo se utilizan los términos siguientes:

atributo corto

designa un atributo que tiene uno de los tipos de datos siguientes: SMALLINT, INTEGER, BIGINT, REAL, DOUBLE, FLOAT, DATE o TIME. También comprende los tipos diferenciados o tipos de referencia basados en esos tipos.

atributo no corto

designa un atributo perteneciente a cualquiera de los tipos de datos restantes o a los tipos diferenciados basados en esos tipos de datos.

El sistema calcula la longitud "inline" por omisión de este modo:

1. Determina las necesidades adicionales de espacio para atributos no cortos, mediante la fórmula siguiente:

$$\text{espacio_para_atributos_no_cortos} = \text{SUM}(\text{longitudAtributo} + n)$$

CREATE TYPE (Estructurado)

n está definido así:

- 0 bytes para atributos de tipo estructurado anidado
- 2 bytes para atributos que no son de LOB
- 9 bytes para atributos de LOB

longitudAtributo está basado en el tipo de datos especificado para el atributo, tal como se describe en la Tabla 27.

2. Calcula la longitud total "inline" por omisión, mediante esta fórmula:

$longitud_por_omisión(tipo_estructurado) = (número_de_atributos * 10) + 32 + espacio_para_atributos_no_cortos$

número_de_atributos es el número total de atributos del tipo estructurado, incluidos los atributos que se heredan del supertipo del tipo. En cambio, *número_de_atributos* no incluye ningún atributo definido para cualquier subtipo del *tipo_estructurado*.

Tabla 27. Contajes de bytes para los tipos de datos de atributos

Tipo de datos de atributo	Contaje de bytes
DECIMAL	Parte entera de $(p/2)+1$, donde p es la precisión
CHAR (n)	n
VARCHAR (n)	n
GRAPHIC (n)	$n * 2$
VARGRAPHIC (n)	$n * 2$
TIMESTAMP	10
DATALINK(n)	$n + 54$

Tabla 27. Contajes de bytes para los tipos de datos de atributos (continuación)

Tipo LOB	Cada atributo LOB tiene un descriptor de LOB, en la instancia del tipo estructurado, que apunta a la ubicación del valor real. El tamaño del descriptor varía de acuerdo con la longitud máxima definida para el atributo LOB	
	Longitud máxima de LOB	Tamaño del descriptor de LOB
	1 024	72
	8 192	96
	65 536	120
	524 000	144
	4 190 000	168
	134 000 000	200
	536 000 000	224
	1 070 000 000	256
	1 470 000 000	280
2 147 483 647	316	
Tipo diferenciado	Longitud del tipo fuente del tipo diferenciado	
Tipo de referencia	Longitud del tipo de datos incorporado en el que está basado el tipo de referencia.	
Tipo estructurado	longitud_inline(<i>tipo_atributo</i>)	

WITHOUT COMPARISONS

Indica que no se da soporte a funciones de comparación para las instancias del tipo estructurado.

NOT FINAL

Indica que el tipo estructurado puede utilizarse como supertipo.

MODE DB2SQL

Esta cláusula es obligatoria y permite la invocación directa de la función constructora para el tipo.

WITH FUNCTION ACCESS

Indica que todos los métodos del tipo y de sus subtipos, incluidos los métodos que se creen en el futuro, se pueden acceder utilizando la notación funcional. Esta cláusula sólo puede especificarse para el tipo raíz de una jerarquía de tipos estructurados (la cláusula UNDER no se especifica) (SQLSTATE 42613). Esta cláusula se proporciona para permitir el uso de la notación funcional para aquellas aplicaciones que prefieren esta forma de notación respecto a la notación de invocación de método.

REF USING *tipo-rep*

Define el tipo de datos incorporados utilizados como representación (tipo

CREATE TYPE (Estructurado)

de datos subyacente) para el tipo de referencia de este tipo estructurado y de todos sus subtipos. Esta cláusula sólo puede especificarse para el tipo raíz de una jerarquía de tipos estructurados (la cláusula UNDER no se especifica) (SQLSTATE 42613). El *tipo-rep* no puede ser un LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB, DATALINK ni un tipo estructurado, y debe tener una longitud de 255 bytes como máximo (SQLSTATE 42613).

Si esta cláusula no se especifica para el tipo raíz de una jerarquía de tipos estructurados, entonces se supone REF USING VARCHAR(16) FOR BIT DATA.

CAST (SOURCE AS REF) WITH *nombrefunc1*

Define el nombre de la función, generada por el sistema, que convierte un valor cuyo tipo de datos es *tipo-rep* al tipo de referencia de este tipo estructurado. No se debe especificar un nombre de esquema como parte de *nombrefunc1* (SQLSTATE 42601). La función de conversión se crea en el mismo esquema que el tipo estructurado. Si no se especifica la cláusula, el valor por omisión para *nombrefunc1* es *nombre-tipo* (el nombre del tipo estructurado). El esquema no debe contener ya una signatura de función que coincida con *nombrefunc1(tipo-rep)*.

CAST (REF AS SOURCE) WITH *nombrefunc2*

Define el nombre de la función, generada por el sistema, que convierte un valor de tipo de referencia para este tipo estructurado al tipo de datos *tipo-rep*. No se debe especificar un nombre de esquema como parte de *nombrefunc2* (SQLSTATE 42601). La función de conversión se crea en el mismo esquema que el tipo estructurado. Si no se especifica la cláusula, el valor por omisión para *nombrefunc2* es *tipo-rep* (el nombre del tipo de representación).

especificación-método

Define los métodos correspondientes a este tipo. Para utilizar un método es necesario proporcionarle un cuerpo mediante una sentencia CREATE METHOD (SQLSTATE 42884).

nombre-método

Designa el método que se está definiendo. Debe ser un identificador SQL no calificado (SQLSTATE 42601). El nombre de método está calificado implícitamente por el esquema utilizado para CREATE TYPE.

Algunos nombres utilizados como palabras clave en predicados están reservados para su uso por el sistema, y no pueden utilizarse como *nombre-método* (SQLSTATE 42939). Estos nombres son SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH y los operadores de comparación que se describen en el apartado “Predicado básico” en la página 213.

En general, puede utilizarse el mismo nombre para más de un método si existe alguna diferencia en la signatura de los métodos.

nombre-parámetro

Designa el nombre del parámetro. Este nombre no puede ser SELF, que es el nombre del parámetro sujeto implícito de un método (SQLSTATE 42734). Si método es un método SQL, todos sus parámetros deben tener nombres (SQLSTATE 42629).

tipo-datos2

Especifica el tipo de datos de cada parámetro. Debe especificarse una entrada de la lista para cada parámetro que el método espera recibir. No se permiten más de 90 parámetros, incluido el parámetro implícito SELF. Si se excede este límite, se produce un error (SQLSTATE 54023).

Se pueden utilizar las especificaciones y abreviaturas de tipos de datos SQL que puedan especificarse como tipo-columna en una sentencia CREATE TABLE y que tengan una correspondencia en el lenguaje utilizado para escribir la método. Consulte las secciones específicas del lenguaje del manual "Application Development Guide" para conocer detalles sobre la correlación entre tipos de datos SQL y tipos de datos de lenguaje principal con respecto a las funciones y métodos definidos por el usuario.

Nota: Si el tipo de datos SQL en cuestión es un tipo estructurado, no existe una correlación por omisión con un tipo de datos de lenguaje principal. Se debe utilizar una función de transformación definida por el usuario para crear una correlación entre el tipo estructurado y el tipo de datos de lenguaje principal.

DECIMAL (y NUMERIC) no son válidos con LANGUAGE C y OLE (SQLSTATE 42815). Para conocer alternativas a la utilización de DECIMAL, consulte el manual *Application Development Guide*.

Se puede especificar REF, pero no tiene un ámbito definido. Dentro del cuerpo del método, se puede utilizar un tipo de referencia en una expresión de vía de acceso sólo si primero se convierte el tipo para que tenga un ámbito. Similarmente, una referencia devuelta por un método se puede utilizar en una expresión de vía de acceso sólo si primero se convierte para que tenga un ámbito.

AS LOCATOR

Para los tipos LOB o tipos diferenciados basados en un tipo LOB, se puede añadir la cláusula AS LOCATOR. Esto indica que se debe pasar un localizador de LOB al método, en lugar de pasarle

CREATE TYPE (Estructurado)

el valor real. Esto reduce considerablemente el número de bytes que se pasan al método y puede también mejorar el rendimiento, especialmente cuando el método sólo necesite unos pocos bytes. La utilización de localizadores de LOB está descrita en el manual *Application Development Guide*.

Se produce un error (SQLSTATE 42601) si se especifica AS LOCATOR para un tipo que no sea LOB o para un tipo diferenciado basado en un LOB.

Si el método es FENCED o si LANGUAGE es SQL, no se puede especificar la cláusula AS LOCATOR (SQLSTATE 42613).

RETURNS

Esta cláusula obligatoria identifica el resultado del método.

tipo-datos3

Especifica el tipo de datos del resultado del método. En este caso, son válidas las mismas consideraciones que para los parámetros de métodos, descritas anteriormente bajo *tipo-datos2*.

AS LOCATOR

Para tipos LOB o tipos diferenciados que están basados en tipos LOB, se puede añadir la cláusula AS LOCATOR. Esto indica que se debe pasar un localizador de LOB desde el método, en lugar del valor real.

Se produce un error (SQLSTATE 42601) si se especifica AS LOCATOR para un tipo que no sea LOB o para un tipo diferenciado basado en un LOB.

Si el método es FENCED o si LANGUAGE es SQL, no se puede especificar la cláusula AS LOCATOR (SQLSTATE 42613).

tipo-datos4 **CAST FROM** *tipo-datos5*

Especifica el tipo de datos del resultado del método.

Esta cláusula se utiliza para devolver a la sentencia invocadora un tipo de datos diferente que el tipo de datos devuelto por el método. El *tipo-datos5* debe ser convertible al parámetro *tipo-datos4*. Si no es convertible, se produce un error (SQLSTATE 42880).

Debido a que la longitud, precisión o escala de *tipo-datos4* puede inferirse de *tipo-datos5*, no es necesario (pero está permitido) especificar la longitud, precisión o escala de los tipos parametrizados especificados para *tipo-datos4*. En su lugar, pueden utilizarse paréntesis vacíos (por ejemplo, VARCHAR()). No puede utilizarse FLOAT() (SQLSTATE 42601), pues el valor del parámetro indica tipos de datos diferentes (REAL o DOUBLE).

No puede especificarse un tipo diferenciado como tipo para tipo-datos5 (SQLSTATE 42815).

La operación de conversión del tipo de datos también está sujeta a comprobaciones durante la ejecución que pueden dar lugar a errores de conversión.

AS LOCATOR

Para tipos LOB o tipos diferenciados que están basados en tipos LOB, se puede añadir la cláusula AS LOCATOR. Esto indica que se debe pasar un localizador de LOB desde el método, en lugar del valor real.

Se produce un error (SQLSTATE 42601) si se especifica AS LOCATOR para un tipo que no sea LOB o para un tipo diferenciado basado en un LOB.

Si el método es FENCED o si LANGUAGE es SQL, no se puede especificar la cláusula AS LOCATOR (SQLSTATE 42613).

SPECIFIC *nombre-específico*

Proporciona un nombre exclusivo para la instancia del método que se está definiendo. Este nombre específico puede utilizarse al crear el cuerpo del método o al eliminar el método. No puede utilizarse nunca para invocar el método. La forma no calificada de *nombre-específico* es un identificador SQL (cuya longitud máxima es 18). La forma calificada es un nombre de esquema seguido de un punto y un identificador SQL. El nombre, incluido el calificador implícito o explícito, no debe identificar otro nombre de método específico que exista en el servidor de aplicaciones; de lo contrario se produce un error (SQLSTATE 42710).

El *nombre-específico* puede ser el mismo que un *nombre-método* existente.

Si no se especifica ningún calificador, se emplea el calificador que se utilizó para *nombre-tipo*. Si se especifica un calificador, debe ser el mismo que el calificador explícito o implícito de *nombre-tipo*, de lo contrario se produce un error (SQLSTATE 42882).

Si no se especifica el *nombre-específico*, el gestor de bases de datos genera un nombre exclusivo. El nombre exclusivo es SQL seguido de una indicación de fecha y hora en forma de serie caracteres: SQLaammddhhmssxxx.

SELF AS RESULT

Identifica el método como método conservador del tipo, lo que significa lo siguiente:

- El tipo de retorno declarado debe ser el mismo que el tipo sujeto declarado (SQLSTATE 428EQ).

CREATE TYPE (Estructurado)

- Cuando una sentencia de SQL se compila y su resolución da un tipo conservador del método, el tipo estático del resultado del método es el mismo que el tipo estático del argumento sujeto.
- Este método se debe implantar de manera que el tipo dinámico del resultado sea el mismo que el tipo dinámico del argumento sujeto (SQLSTATE 2200G) y el resultado no puede tampoco ser nulo (SQLSTATE 22004).

características-rutina-SQL

Especifica las características del cuerpo del método que se definirán para este tipo utilizando CREATE METHOD.

LANGUAGE SQL

Esta cláusula se utiliza para indicar que el método está escrito en SQL mediante una sola sentencia RETURN. El cuerpo del método se especifica utilizando la sentencia CREATE METHOD.

NOT DETERMINISTIC o DETERMINISTIC

Esta cláusula opcional especifica si el método siempre devuelve los mismos resultados para valores de argumento determinados (DETERMINISTIC) o si el método depende de ciertos valores de estado que afectan a los resultados (NOT DETERMINISTIC). Es decir, un método DETERMINISTIC siempre debe devolver el mismo resultado para invocaciones sucesivas con entradas de datos idénticas. Se evitan las optimizaciones que aprovechan el hecho de que entradas idénticas siempre produzcan los mismos resultados especificando NOT DETERMINISTIC. Se debe especificar NOT DETERMINISTIC, de forma explícita o implícita, si el cuerpo del método accede a un registro especial, o invoca otra rutina no determinista (SQLSTATE 428C2).

NO EXTERNAL ACTION o EXTERNAL ACTION

Esta cláusula opcional especifica si el método realiza alguna acción que cambia el estado de un objeto no gestionado por el gestor de bases de datos. Para evitar las optimizaciones basadas en que el método no produce ningún efecto externo, se especifica EXTERNAL ACTION. Por ejemplo: enviar un mensaje, hacer sonar una alarma o grabar un registro en un archivo.

READS SQL DATA o CONTAINS SQL

Indica qué tipo de sentencias de SQL se pueden ejecutar. Debido a que la sentencia de SQL soportada es la sentencia RETURN, la distinción está relacionada con el hecho de si la expresión es una subconsulta o no.

READS SQL DATA

Indica que el método puede ejecutar sentencias de SQL que no modifican datos SQL (SQLSTATE 42985). No se pueden utilizar apodos en la sentencia de SQL (SQLSTATE 42997).

CONTAINS SQL

Indica que el método puede ejecutar sentencias de SQL que no leen ni modifican datos SQL (SQLSTATE 42985).

CALLED ON NULL INPUT

Esta cláusula opcional indica que debe invocarse el método definido por el usuario aunque contenga argumentos nulos. Puede devolver un valor nulo o un valor normal (no nulo). Sin embargo, corresponde al método comprobar si hay valores de argumento nulos.

El valor NULL CALL puede utilizarse como sinónimo de CALLED ON NULL INPUT por razones de compatibilidad.

características-rutina-externa

LANGUAGE

Esta cláusula obligatoria se emplea para especificar el convenio de la interfaz de lenguaje para el está escrito el cuerpo del método definido por el usuario.

C Esto significa que el gestor de bases de datos invocará el método definido por el usuario como si fuera una función escrita en C. El método definido por el usuario debe cumplir el convenio de invocación y enlace del lenguaje C, tal como está definido por el prototipo C estándar de ANSI.

JAVA

Esto significa que el gestor de bases de datos invocará el método definido por el usuario como si fuera un método de una clase Java.

OLE

Esto significa que el gestor de bases de datos invocará el método definido por el usuario como si fuera un método expuesto por un objeto de automatización OLE. El método debe ajustarse a los tipos de datos de automatización y al mecanismo de invocación de OLE, tal como se describe en el manual "OLE Automation Programmer's Reference".

Sólo se da soporte a LANGUAGE OLE para métodos definidos por el usuario almacenados en Sistemas operativos Windows de 32 bits.

PARAMETER STYLE

Esta cláusula se utiliza para especificar los convenios utilizados para pasar parámetros al método y obtener el resultado del método.

DB2SQL

Se utiliza para especificar los convenios para pasar parámetros a métodos externos y obtener su resultado; estos métodos cumplen los convenios de invocación y enlace del lenguaje C o son

CREATE TYPE (Estructurado)

métodos expuestos por objetos de automatización OLE. Esta opción debe especificarse cuando se utiliza LANGUAGE C o LANGUAGE OLE.

DB2GENERAL

Se utiliza para especificar los convenios para pasar parámetros a métodos externos y obtener sus resultados; estos métodos están definidos como tales en una clase Java. Esto sólo puede especificarse cuando se utiliza LANGUAGE JAVA.

El valor DB2GENRL puede utilizarse como sinónimo de DB2GENERAL.

Consulte el manual *Application Development Guide* para conocer detalles sobre el pase de parámetros.

DETERMINISTIC o NOT DETERMINISTIC

Esta cláusula opcional especifica si el método siempre devuelve los mismos resultados para valores de argumento determinados (DETERMINISTIC) o si el método depende de ciertos valores de estado que afectan a los resultados (NOT DETERMINISTIC). Es decir, un método DETERMINISTIC siempre debe devolver el mismo resultado para invocaciones sucesivas con entradas de datos idénticas. Se evitan las optimizaciones que aprovechan el hecho de que entradas idénticas siempre produzcan los mismos resultados especificando NOT DETERMINISTIC.

Un ejemplo de método NOT DETERMINISTIC sería un método que devuelve al azar un número de identificación de un empleado de un departamento. Un ejemplo de método DETERMINISTIC sería un método que calcula el área de un polígono.

FENCED o NOT FENCED

Esta cláusula especifica si el método se considera "seguro" (NOT FENCED) o no (FENCED) para ejecutarse en el proceso o espacio de direcciones del entorno operativo del gestor de bases de datos.

Si un método se registra como FENCED, el gestor de bases de datos aísla los recursos internos de base de datos (por ejemplo, almacenamientos intermedios de datos) para que el método no pueda acceder a ellos. La mayoría de los métodos tienen la posibilidad de ejecutarse como FENCED o NOT FENCED. En general, un método que se ejecute como FENCED no funcionará tan bien como otro método similar que se ejecute como NOT FENCED.

Nota: La utilización de NOT FENCED para métodos no probados debidamente puede comprometer la integridad de DB2. DB2 toma algunas precauciones para muchos de los errores involuntarios más habituales, pero no puede garantizar la

integridad total de los datos cuando se utilizan métodos NOT FENCED definidos por el usuario.

Observe que, aunque el uso de FENCED protege mejor la integridad de la base de datos, un método definido como FENCED que no se haya codificado, revisado y probado debidamente puede también causar un error involuntario en DB2.

Normalmente, la mayoría de los métodos se pueden ejecutar como FENCED o NOT FENCED. Un método definido con LANGUAGE OLE sólo admite la especificación FENCED (SQLSTATE 42613).

Si el método es FENCED, no se puede especificar la cláusula AS LOCATOR (SQLSTATE 42613).

Para cambiar de FENCED a NOT FENCED, es necesario volver a registrar el método (se debe eliminar y después volverlo a crear).

Para registrar un método como NOT FENCED, es necesaria la autorización SYSADM, la autorización DBADM o una autorización especial (CREATE_NOT_FENCED).

RETURNS NULL ON NULL INPUT o CALLED ON NULL INPUT

Esta cláusula opcional sirve para evitar la invocación de un método externo si cualquiera de los argumentos no sujetos es nulo.

Si se especifica RETURNS NULL ON NULL INPUT y al ejecutar el método alguno de sus argumentos es nulo, no se invoca el método y el resultado es el valor nulo.

Si se especifica CALLED ON NULL INPUT, se invoca el método con independencia de si hay argumentos nulos. Puede devolver un valor nulo o un valor normal (no nulo). Sin embargo, corresponde al método comprobar si hay valores de argumento nulos.

El valor NULL CALL puede utilizarse como sinónimo de CALLED ON NULL INPUT para mantener la compatibilidad con versiones anteriores. De manera similar, puede utilizarse NOT NULL CALL como sinónimo de RETURNS NULL ON NULL INPUT.

Existen dos casos para los que no se tiene en cuenta esta especificación:

- Si el argumento sujeto es nulo. En este caso el método no se ejecuta y el resultado es nulo.
- Si el método está definido para no tener parámetros. En este caso la condición de argumento nulo no puede producirse.

CREATE TYPE (Estructurado)

NO SQL

Esta cláusula obligatoria indica que el método no puede emitir ninguna sentencia de SQL. Si las emite, se genera un error durante la ejecución (SQLSTATE 38502).

EXTERNAL ACTION o NO EXTERNAL ACTION

Esta cláusula opcional especifica si el método realiza alguna acción que cambia el estado de un objeto no gestionado por el gestor de bases de datos. Para evitar las optimizaciones basadas en que el método no produce ningún efecto externo, se especifica EXTERNAL ACTION.

NO SCRATCHPAD o SCRATCHPAD *longitud*

Esta cláusula opcional especifica si debe proporcionarse una memoria de trabajo para un método externo. Es muy recomendable que los métodos sean reentrantes, por lo que una memoria de trabajo proporciona un medio para que el método "guarde el estado" entre una llamada y la siguiente.

Si se especifica SCRATCHPAD, cuando se efectúa la primera invocación del método definido por el usuario, se asigna memoria para que el método externo utilice una memoria de trabajo. Esta memoria de trabajo tiene las características siguientes:

- *longitud*, si se especifica, define el tamaño en bytes de la memoria de trabajo; este valor debe estar comprendido entre 1 y 32.767 (SQLSTATE 42820). El valor por omisión es 100.
- El valor de inicialización consta sólo de ceros hexadecimales.
- Su ámbito es la sentencia de SQL. Existe una memoria de trabajo para cada referencia al método externo contenida en la sentencia de SQL.

Por tanto, si el método X de la sentencia siguiente se define con la palabra clave SCRATCHPAD, se asignarán tres memorias de trabajo.

```
SELECT A, X..(A) FROM TABLEB
WHERE X..(A) > 103 OR X..(A) < 19
```

Si se especifica ALLOW PARALLEL o se toma por omisión, el ámbito es diferente del anterior. Si el método se ejecuta en varias particiones, se asigna una memoria de trabajo en cada partición donde se procesa el método, una para cada referencia al método contenida en la sentencia de SQL. De manera similar, si la consulta se ejecuta con el paralelismo de intrapartición habilitado, pueden asignarse más de tres memorias de trabajo.

La memoria de trabajo es permanente. Su contenido se conserva entre una llamada al método externo y la siguiente. Los cambios hechos en la memoria de trabajo por el método externo en una llamada seguirán

allí en la llamada siguiente. El gestor de bases de datos inicializa las memorias de trabajo al principio de la ejecución de cada sentencia de SQL. El gestor de bases de datos puede restaurar las memorias de trabajo al comienzo de la ejecución de cada subconsulta. El sistema emite una llamada final antes de restaurar una memoria de trabajo si se especifica la opción FINAL CALL.

La memoria de trabajo puede utilizarse como punto central de los recursos del sistema (por ejemplo, la memoria) que podría adquirir el método externo. El método podría adquirir la memoria en la primera llamada, conservar su dirección en la memoria de trabajo y acceder a ella en llamadas posteriores.

En el caso en que se adquiere el recurso del sistema, también debe especificarse la palabra clave FINAL CALL; esto provoca una llamada especial en el final de la sentencia para permitir que el método externo libere los recursos del sistema adquiridos.

Si se especifica SCRATCHPAD, en cada invocación del método definido por el usuario se pasa un argumento adicional al método externo que indica la dirección de la memoria de trabajo.

Si se especifica NO SCRATCHPAD, no se asignará ni pasará ninguna memoria de trabajo al método externo.

NO FINAL CALL o FINAL CALL

Esta cláusula opcional especifica si debe realizarse una llamada final a un método externo. La finalidad de esta llamada final es permitir que el método externo libere los recursos que ha adquirido del sistema. Junto con la palabra clave SCRATCHPAD, esta cláusula puede ser útil en situaciones donde el método externo adquiere recursos del sistema, tales como memoria, y los fija en la memoria de trabajo.

Si se especifica FINAL CALL, durante la ejecución se pasa al método externo un argumento adicional que especifica el tipo de llamada. Los tipos de llamadas son:

- Llamada normal: Se pasan argumentos SQL y se espera la devolución de un resultado.
- Primera llamada: es la primera llamada al método externo para esta referencia específica al método contenida en esta sentencia de SQL específica. La primera llamada es una llamada normal.
- Llamada final: es una llamada final al método externo para permitir que el método libere recursos. La llamada final no es una llamada normal. Esta llamada final tiene lugar en los momentos siguientes:

CREATE TYPE (Estructurado)

- Fin de sentencia: Esta situación se produce cuando se cierra el cursor, en el caso de sentencias basadas en cursores, o cuando la sentencia termina su ejecución de otra manera.
- Fin de transacción: Este caso se produce cuando no se da un fin de sentencia normal. Por ejemplo, cuando por alguna razón el código de una aplicación elude el cierre del cursor.

Si se produce una operación de confirmación mientras está abierto un cursor definido como `WITH HOLD`, se realiza una llamada final en el cierre subsiguiente del cursor o al final de la aplicación.

Si se especifica `NO FINAL CALL`, no se pasa al método externo ningún argumento que especifique el tipo de llamada, y no se realiza ninguna llamada final.

ALLOW PARALLEL o DISALLOW PARALLEL

Esta cláusula opcional especifica si, para una referencia individual al método, puede paralelizarse la invocación del método. En general, las invocaciones de la mayoría de los métodos escalares pueden paralelizarse, pero puede haber métodos (tales como los que dependen de una sola copia de una memoria de trabajo) que no puedan. Si se especifica `ALLOW PARALLEL` o `DISALLOW PARALLEL` para un método, DB2 aceptará esta especificación.

Deben tenerse en cuenta las cuestiones siguientes al determinar qué palabra clave es la adecuada para el método:

- ¿Son todas las invocaciones del método completamente independientes las unas de las otras? En caso afirmativo, especifique `ALLOW PARALLEL`.
- ¿Se actualiza la memoria de trabajo con cada invocación del método, proporcionando valores que son de interés para la siguiente invocación (el incremento de un contador, por ejemplo)? En caso afirmativo, especifique `DISALLOW PARALLEL` o acepte el valor por omisión.
- ¿Realiza el método alguna acción externa que deba producirse en una sola partición? En caso afirmativo, especifique `DISALLOW PARALLEL` o acepte el valor por omisión.
- ¿Se utiliza la memoria de trabajo, pero requiere realizar algún proceso de inicialización costoso un número mínimo de veces? En caso afirmativo, especifique `ALLOW PARALLEL`.

En cualquier caso, el cuerpo de cada método externo debe estar en un directorio que sea accesible en todas las particiones de la base de datos.

El diagrama de sintaxis indica que el valor por omisión es ALLOW PARALLEL. Sin embargo, el valor por omisión es DISALLOW PARALLEL si se especifica una o más de las opciones siguientes en la sentencia:

- NOT DETERMINISTIC
- EXTERNAL ACTION
- SCRATCHPAD
- FINAL CALL

NO DBINFO o DBINFO

Esta cláusula opcional especifica si se pasará (DBINFO) o no (NO DBINFO) al método cierta información específica conocida por DB2, en forma de argumento adicional de invocación. NO DBINFO es el valor por omisión. DBINFO no puede utilizarse para LANGUAGE OLE (SQLSTATE 42613).

Si se especifica DBINFO, se pasa una estructura al método, la cual contiene la información siguiente:

- Nombre de base de datos - el nombre de la base de datos conectada actualmente.
- ID de aplicación - ID de aplicación exclusivo que se establece para cada conexión con la base de datos.
- ID de autorización de aplicación - el ID de autorización de ejecución de la aplicación, sin tener en cuenta los métodos anidados existentes entre este método y la aplicación.
- Página de códigos - identifica la página de códigos de la base de datos.
- Nombre de esquema - si se dan exactamente las mismas condiciones que para el Nombre de tabla, contiene el nombre del esquema; en otro caso, está en blanco.
- Nombre de tabla - si la referencia a método es la parte derecha de una cláusula SET en una sentencia UPDATE o es un elemento de la lista VALUES de una sentencia INSERT, este dato es el nombre no calificado de la tabla que se está actualizando o insertando; en otro caso, está en blanco.
- Nombre de columna - si se dan las mismas condiciones que para el Nombre de tabla, este dato es el nombre de la columna que se está actualizando o insertando; en otro caso, está en blanco.
- Versión/release de base de datos - identifica la versión, release y nivel de modificación del servidor de bases de datos invocador del método.
- Plataforma - contiene el tipo de plataforma del servidor.

CREATE TYPE (Estructurado)

- Números de columna del resultado del método de tabla - no es aplicable a métodos.

Vea el manual *Application Development Guide* para obtener información detallada sobre la estructura y cómo se pasa al método.

Notas

- La creación de un tipo estructurado con un nombre de esquema que todavía no existe dará como resultado la creación implícita de este esquema siempre que el ID de autorización de la sentencia tenga la autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN en el esquema se otorga a PUBLIC.
- Un subtipo estructurado definido sin atributos define un subtipo que hereda todos sus atributos de un supertipo. Si no se especifica una cláusula UNDER ni ningún otro atributo, el tipo es un tipo raíz de una jerarquía de tipos, sin ningún atributo.
- La adición de un nuevo subtipo a una jerarquía de tipos puede causar la invalidación de paquetes. Un paquete puede quedar invalidado si depende de un supertipo del nuevo tipo. Dicha dependencia es el resultado de la utilización de un predicado TYPE o de una especificación TREAT.
- Un tipo estructurado puede tener 4082 atributos como máximo (SQLSTATE 54050).
- Una especificación de método no puede tener la misma signatura que una función (cuando se compara el primer tipo-parámetro de la función con el tipo-sujeto del método).
- Un método MT, cuyo tipo-sujeto es T, está definido para invalidar otro método MS, con el tipo-sujeto S, si se cumplen todas estas condiciones:
 - MT y MS tienen el mismo nombre no calificado y el mismo número de parámetros.
 - T es un subtipo apropiado de S
 - Los tipos de parámetros no sujetos de MT son iguales a los correspondientes tipos de parámetros no sujetos de MS. Observe que esta igualdad se refiere al tipo básico, tal como VARCHAR, sin tener en cuenta la longitud ni la precisión.

Ningún método puede invalidar a otro método ni ser invalidado por él (SQLSTATE 42745). Además, Una función y un método no pueden prevalecer el uno sobre el otro (SQLSTATE 42745). Esto significa que si la función fuera un método con su primer parámetro como sujeto S, no debe invalidar a otro método de ningún supertipo de S, y no debe ser invalidado por otro método de ningún subtipo de S.

- La creación de un tipo estructurado genera automáticamente un conjunto de funciones y métodos que pueden utilizarse con el tipo. Todas las funciones y métodos se generan en el mismo esquema que el tipo

estructurado. Si la signatura de la función o método generados entra en conflicto con o invalida la signatura de una función existente en este esquema, la sentencia falla (SQLSTATE 42710). Las funciones o métodos generados no pueden eliminarse sin eliminar el tipo estructurado (SQLSTATE 42917). Se generan las funciones y métodos siguientes:

– Funciones

- Comparaciones de referencia

Se generan seis funciones de comparación denominadas =, <>, <, <=, >, >= para el tipo de referencia REF(*nombre-tipo*). Cada una de estas funciones toma dos parámetros del tipo REF(*nombre-tipo*) y devuelve el valor de verdadero, falso o desconocido. Los operadores de comparación para REF(*nombre-tipo*) se definen de manera que actúen como los operadores de comparación para el tipo de datos subyacente de REF(*nombre-tipo*).⁸⁶

El ámbito del tipo de referencia no se tiene en cuenta en la comparación.

- Funciones de conversión

Se generan dos funciones de conversión para convertir entre el tipo de referencia generado REF(*nombre-tipo*) y el tipo de datos subyacente de este tipo de referencia.

- El nombre de la función para convertir el tipo subyacente al tipo de referencia es el *nombrefunc1* implícito o explícito.

El formato de esta función es:

```
CREATE FUNCTION nombrefunc1 (tipo-rep)
RETURNS REF(nombre-tipo)
```

...

- El nombre de la función para convertir el tipo de referencia al tipo subyacente del tipo de referencia es el *nombrefunc2* implícito o explícito.

El formato de esta función es:

```
CREATE FUNCTION nombrefunc2 ( REF(nombre-tipo) )
RETURNS tipo-rep ...
```

Para algunos tipos-rep, existen funciones de conversión adicionales generadas con *nombrefunc1* para manejar las conversiones desde constantes.

- Si *tipo-rep* es SMALLINT, la función de conversión adicional generada tiene el formato:

86. Todas las referencias de una jerarquía de tipos tienen el mismo tipo de representación de referencia. Esto permite que se comparen REF(S) y REF(T) siempre que S y T tengan un supertipo común. Dado que la exclusividad de la columna de OID sólo se impone dentro de una jerarquía de tablas, es posible que un valor de REF(T) de una jerarquía de tablas sea "igual" a un valor de REF(T) de otra jerarquía de tablas, aunque hagan referencia a filas diferentes.

CREATE TYPE (Estructurado)

```
CREATE FUNCTION nombrefunc1 (INTEGER)
  RETURNS REF(nombre-tipo)
```

- Si *tipo-rep* es CHAR(*n*), la función de conversión adicional generada tiene el formato:

```
CREATE FUNCTION nombrefunc1 ( VARCHAR(n))
  RETURNS REF(nombre-tipo)
```

- Si *tipo-rep* es GRAPHIC(*n*), la función de conversión adicional generada tiene el formato:

```
CREATE FUNCTION nombrefunc1 (VARGRAPHIC(n))
  RETURNS REF(nombre-tipo)
```

El nombre de esquema del tipo estructurado debe incluirse en la vía de acceso de SQL (consulte el apartado “SET PATH” en la página 1181 o la opción FUNCPATH BIND tal como se describe en el manual *Application Development Guide*) para la utilización satisfactoria de estos operadores y funciones de conversión en las sentencias de SQL.

- Función constructora

La función constructora se genera para permitir la creación de una nueva instancia del tipo. Esta nueva instancia tendrá el valor nulo para todos los atributos del tipo, incluidos los atributos que se heredan de un supertipo.

El formato de la función constructora generada es:

```
CREATE FUNCTION nombre-tipo ( )
  RETURNS nombre-tipo
  ...
```

Si se especifica NOT INSTANTIABLE, no se genera ninguna función constructora. Si el tipo estructurado tiene atributos de tipo DATALINK, la invocación de la función constructora no surte efecto (SQLSTATE 428ED).

- Métodos

- Métodos observadores

Se define un método observador para cada atributo del tipo estructurado. Para cada atributo, el método observador devuelve el tipo del atributo. Si el sujeto es nulo, el método observador devuelve un valor nulo del tipo de atributo.

Por ejemplo, los atributos de una instancia del tipo estructurado ADDRESS se pueden observar utilizando C1..STREET, C1..CITY, C1..COUNTRY y C1..CODE .

La signatura del método observador generado es como si se hubiera ejecutado la sentencia siguiente:


```
CREATE TYPE nombre-tipo
...
METHOD nombre-atributo()
RETURNS tipo-atributo
```

donde *nombre-tipo* es el nombre del tipo estructurado.

- Métodos mutadores

Se define un método mutador preservador del tipo para cada atributo del tipo estructurado. Utilice métodos mutadores para cambiar atributos dentro de una instancia de un tipo estructurado. Para cada atributo, el método mutador devuelve una copia del sujeto modificada por la asignación del argumento al atributo mencionado de la copia.

Por ejemplo, una instancia del tipo estructurado ADDRESS se puede mutar utilizando C1..CODE('M3C1H7'). Si el sujeto es nulo, el método mutador emite un error (SQLSTATE 2202D).

La signatura del método mutador generado es como si se hubiera ejecutado la sentencia siguiente:

```
CREATE TYPE nombre-tipo
...
METHOD nombre-atributo (tipo-atributo)
RETURNS nombre-tipo
```

Si el tipo de datos del atributo es SMALLINT, REAL, CHAR o GRAPHIC, se genera un método mutador adicional para dar soporte a la mutación utilizando constantes:

- Si *tipo-atributo* es SMALLINT, el mutador adicional da soporte a un argumento de tipo INTEGER.
 - Si *tipo-atributo* es REAL, el mutador adicional da soporte a un argumento de tipo DOUBLE.
 - Si *tipo-atributo* es CHAR, el mutador adicional da soporte a un argumento de tipo VARCHAR.
 - Si *tipo-atributo* es GRAPHIC, el mutador adicional da soporte a un argumento de tipo VARGRAPHIC.
- Si el tipo estructurado se utiliza como tipo de columna, la longitud máxima de una instancia del tipo es 1 GB durante la ejecución (SQLSTATE 54049).
- Cuando se crea un nuevo subtipo para un tipo estructurado existente (para utilizarlo como tipo de columna), se deben reexaminar y actualizar según sea necesario las funciones de transformación existentes que dan soporte a los tipos estructurados asociados. Tanto si el nuevo tipo está en la misma jerarquía que un tipo determinado o en la jerarquía de un tipo anidado, es probable que la función de transformación asociada a este tipo deba modificarse para incluir algunos o todos los nuevos atributos originados por el nuevo subtipo. En general, debido a que es el conjunto de las

CREATE TYPE (Estructurado)

funciones de transformación asociadas a un tipo determinado (o jerarquía de tipos) lo que permite que las UDF y aplicaciones cliente accedan al tipo estructurado, las funciones de transformación se deben escribir para dar soporte a TODOS los atributos de una jerarquía compuesta determinada (es decir, incluido el cierre transitivo de todos los subtipos y sus tipos estructurados anidados). Puesto que es el conjunto de funciones de transformación asociado con un tipo determinado (o jerarquía de tipos) el que activa el acceso de UDF y Aplicaciones cliente al tipo estructurado, las funciones de transformación deben escribirse de modo que den soporte a TODOS los atributos de una jerarquía compuesta determinada (esto es, incluido el cierre transitivo de todos los subtipos y sus tipos estructurados anidados).

Ejemplos

Ejemplo 1: Cree un tipo para Department.

```
CREATE TYPE DEPT AS
  (DEPT_NAME  VARCHAR(20),
   MAX_EMPS  INT,
   REF USING INT
  )
MODE DB2SQL
```

Ejemplo 2: Cree una jerarquía de tipos compuesta de un tipo para los empleados y de un subtipo para los directores.

```
CREATE TYPE EMP AS
  (NAME      VARCHAR(32),
   SERIALNUM INT,
   DEPT      REF(DEPT),
   SALARY    DECIMAL(10,2) )
MODE DB2SQL
```

```
CREATE TYPE MGR UNDER EMP AS
  (BONUS     DECIMAL(10,2))
MODE DB2SQL
```

Ejemplo 3: Cree una jerarquía de tipos para direcciones. Las direcciones están destinadas a utilizarse como tipos de columnas. La longitud "inline" no se especifica, por lo que DB2 calculará una longitud por omisión. Dentro de la definición del tipo de dirección se encapsulará un método externo que calcula el grado de proximidad de la dirección con una dirección de entrada proporcionada. El cuerpo de la sentencia se crea mediante la sentencia CREATE METHOD.

```
CREATE TYPE address_t AS
  (STREET     VARCHAR(30),
   NUMBER     CHAR(15),
   CITY       VARCHAR(30),
   STATE      VARCHAR(10))
  NOT FINAL
MODE DB2SQL
  METHOD SAMEZIP (addr address_t)
```

CREATE TYPE (Estructurado)

```
RETURNS INTEGER
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
NO EXTERNAL ACTION

METHOD DISTANCE (address_t)
RETURNS FLOAT
LANGUAGE C
DETERMINISTIC
PARAMETER STYLE DB2SQL
NO SQL
NO EXTERNAL ACTION
```

```
CREATE TYPE germany_addr_t UNDER address_t AS
(FAMILY_NAME VARCHAR(30))
NOT FINAL
MODE DB2SQL
```

```
CREATE TYPE us_addr_t UNDER address_t AS
(ZIP VARCHAR(10))
NOT FINAL
MODE DB2SQL
```

Ejemplo 4: Creación de un tipo que tenga atributos de tipo estructurado anidados.

```
CREATE TYPE PROJECT AS
(PROJ_NAME VARCHAR(20),
 PROJ_ID INTEGER,
 PROJ_MGR MGR,
 PROJ_LEAD EMP,
 LOCATION ADDR_T,
 AVAIL_DATE DATE)
MODE DB2SQL
```

CREATE TYPE MAPPING

CREATE TYPE MAPPING

La sentencia CREATE TYPE MAPPING crea una correlación entre estos tipos de datos:

- Un tipo de datos de una columna de una tabla o una vista de la fuente de datos que se va a definir en una base de datos federada
- Un tipo de datos correspondiente que ya está definido en la base de datos federada.

La correlación puede asociar el tipo de datos de la base de datos federada con un tipo de datos de (1) una fuente de datos especificada o de (2) un rango de fuentes de datos; por ejemplo, todas las fuentes de datos de un tipo y versión en particular.

Una correlación de tipos de datos sólo debe crearse si una existente no es adecuada.

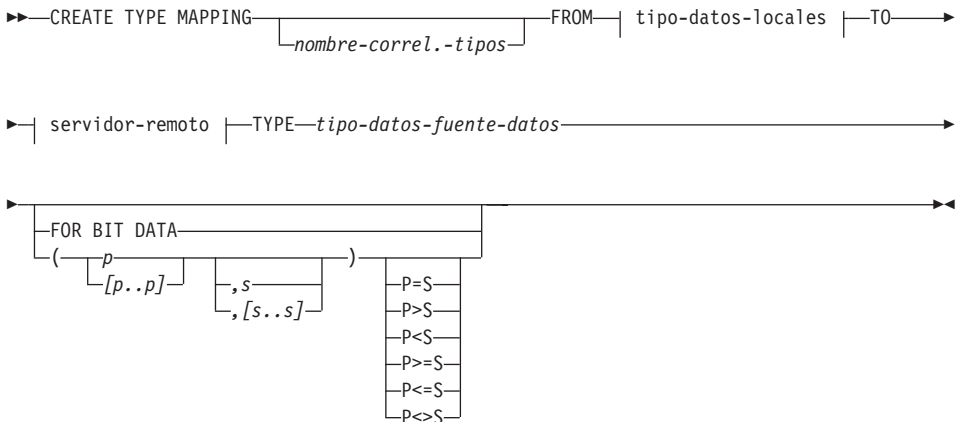
Invocación

Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

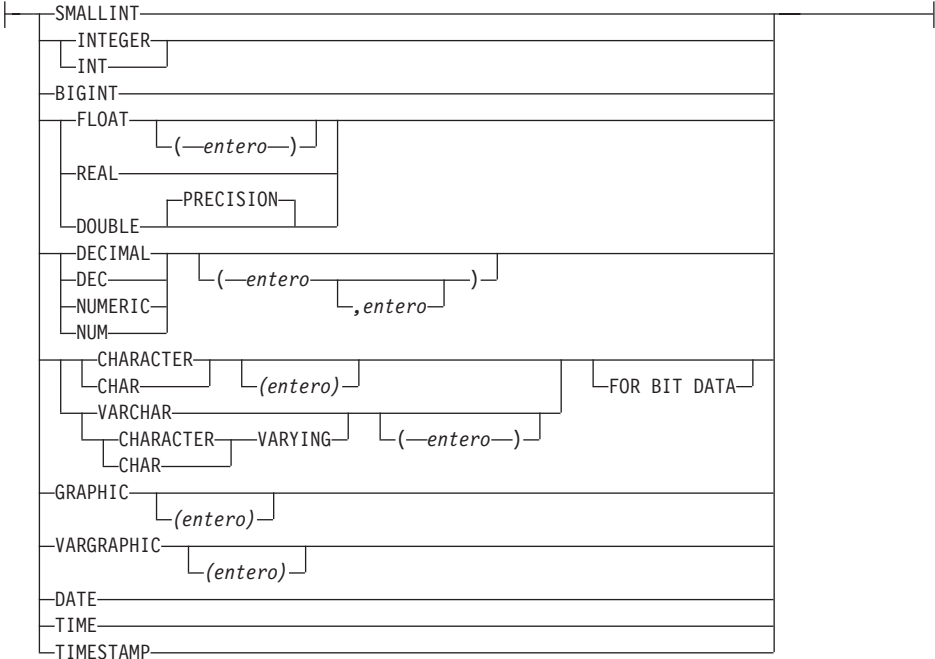
Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben tener la autorización SYSADM o DBADM.

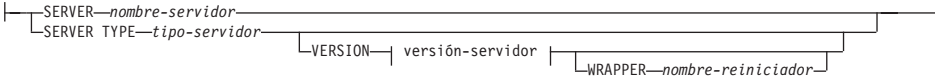
Sintaxis



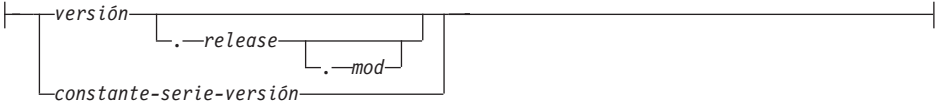
tipo-datos-locales:



servidor-remoto:



versión-servidor:



Descripción

nombre-correlación-tipos

Designa la correlación de tipos de datos. El nombre no debe identificar ninguna correlación de datos que ya esté descrita en el catálogo. Se genera un nombre exclusivo si no se especifica nombre-correlación-tipos.

CREATE TYPE MAPPING

tipo-datos-locales

Identifica un tipo de datos que se define en una base de datos federada. Si se especifica *tipo-datos-locales* sin ningún nombre de esquema, el nombre de tipo se resuelve buscando en los esquemas especificados en la vía de acceso SQL (definida por la opción de preproceso FUNCPATH en el caso del SQL estático y por el registro CURRENT PATH en el caso del SQL dinámico). Si la longitud o precisión (y escala) no se especifican para el *tipo-datos-local*, los valores se determinan a partir del *tipo-datos-fuente*.

El *tipo-datos-local* no puede ser LONG VARCHAR, LONG VARGRAPHIC, DATALINK, un tipo de objeto grande (LOB) ni un tipo definido por el usuario (SQLSTATE 42806).

SERVER *nombre-servidor*

Nombra la fuente de datos para la que se define *tipo-datos-fuente-datos*.

SERVER TYPE *tipo-servidor*

Identifica el tipo de datos para el que se define *tipo-datos-fuente-datos*.

VERSION

Identifica la versión de la fuente de datos para la que se define el *tipo-datos-fuente-datos*.

versión

Especifica el número de versión. *versión* debe ser un entero.

release

Especifica el número del release de la versión indicada por *versión*. *release* debe ser un entero.

mod

Especifica el número de la modificación del release indicado por *release*. *mod* debe ser un entero.

constante-serie-versión

Especifica la designación completa de la versión. La *constante-serie-versión* puede ser un solo valor (por ejemplo, '8i'); o puede ser los valores concatenados de *versión*, *release* y, si se aplica, *mod* (por ejemplo, '8.0.3').

WRAPPER *nombre-reiniciador*

Especifica el nombre del reiniciador que el servidor federado utiliza para interactuar con las fuentes de datos del tipo y versión indicados por *tipo-servidor* y *versión-servidor*.

TYPE *tipo-datos-fuente-datos*

Especifica el tipo de datos de la fuente de datos que se está correlacionando con el *tipo-datos-local*. Si *tipo-datos-fuente-datos* está calificado por un nombre de esquema en la fuente de datos, está permitido pero no es necesario, especificar este calificador.

El *tipo-datos-fuente-datos* debe ser un tipo de datos incorporado. No están permitidos los tipos definidos por el usuario. Si el tipo tiene un formato corto y otro largo (por ejemplo, CHAR y CHARACTER), debe especificarse el formato corto.

- p* Para un tipo de datos decimal, *p* especifica el número máximo de dígitos que puede tener un valor. Para todos los demás tipos de datos de caracteres, *p* especifica el número máximo de caracteres que puede tener un valor. *p* debe ser válido con respecto al tipo de datos (SQLSTATE 42611). Si no se especifica *p* y el tipo de datos lo necesita, el sistema determinará la mejor opción.

[*p..p*]

Para un tipo de datos decimal, [*p..p*] especifica el número mínimo y máximo de dígitos que puede tener un valor. Para los demás tipos de datos de caracteres, [*p..p*] especifica el número mínimo y máximo de caracteres que puede tener un valor. En todos los casos, el máximo debe ser igual o sobrepasar el mínimo; y ambos números deben ser válidos con respecto al tipo de datos (SQLSTATE 42611).

- s* Para un tipo de datos decimal, *s* especifica el número máximo permitido de dígitos a la derecha de la coma decimal. Este número debe ser válido con respecto al tipo de datos (SQLSTATE 42611). Si no se especifica ningún número y el tipo de datos necesita uno, el sistema determinará la mejor opción.

[*s..s*]

Para un tipo de datos decimal, [*s..s*] especifica el número mínimo y máximo de dígitos permitidos a la derecha de la coma decimal. El máximo debe ser igual o exceder del mínimo y ambos números deben ser válidos con respecto al tipo de datos (SQLSTATE 42611).

P [operando] S

Para un tipo de datos decimal, P [operando] S especifica una comparación entre la precisión máxima permitida y el número máximo de dígitos permitidos a la derecha de la coma decimal. Por ejemplo, el operando = indica que la precisión máxima permitida y el número máximo de dígitos permitidos en la fracción decimal son iguales. Especifique P [operando] S solamente si el nivel de comprobación que impone es necesario.

FOR BIT DATA

Indica si *tipo-datos-fuente-datos* es para los datos de bits. Estas palabras clave son necesarias si la columna del tipo de la fuente de datos contiene valores binarios. El gestor de bases de datos determinará este atributo si no se especifica en un tipo de datos de caracteres.

Notas

Una sentencia CREATE TYPE MAPPING de una unidad de trabajo (UOW) determinada no puede procesarse bajo ninguna de las condiciones siguientes:

CREATE TYPE MAPPING

- La sentencia hace referencia a una sola fuente de datos y la UOW ya incluye una sentencia SELECT que hace referencia a un apodo para una tabla o una vista de esta fuente de datos.
- La sentencia hace referencia a una categoría de fuentes de datos (por ejemplo, todas las fuentes de datos de un tipo y versión específicos) y la UOW ya incluye una sentencia SELECT que hace referencia a un apodo para una tabla o una vista de una de estas fuentes de datos.

Ejemplos

Ejemplo 1: Cree una correlación entre SYSIBM.DATE y el tipo de datos DATE de Oracle y en todas las fuentes de datos Oracle.

```
CREATE TYPE MAPPING MY_ORACLE_DATE  
FROM SYSIBM.DATE  
TO SERVER TYPE ORACLE  
TYPE DATE
```

Ejemplo 2: Cree una correlación entre SYSIBM.DECIMAL(10,2) y el tipo de datos NUMBER([10..38],2) de Oracle en la fuente de datos ORACLE1.

```
CREATE TYPE MAPPING MY_ORACLE_DEC  
FROM SYSIBM.DECIMAL(10,2)  
TO SERVER ORACLE1  
TYPE NUMBER([10..38],2)
```


CREATE USER MAPPING

La sentencia CREATE USER MAPPING define una correlación entre un ID de autorización que utiliza una base de datos federada y el ID de autorización y la contraseña que se han de utilizar en una fuente de datos especificada.

Invocación

Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

Autorización

Si el ID de autorización de la sentencia es diferente del nombre de autorización que se está correlacionando con la fuente de datos, el ID de autorización debe incluir la autorización SYSADM o DBADM. De lo contrario, si el ID de autorización y el nombre de autorización coinciden, no es necesario ningún privilegio ni ninguna autorización.

Sintaxis

```

▶▶ CREATE USER MAPPING FOR nombre-autorización SERVER nombre-servidor
    USER
▶
▶ OPTIONS (
    ADD nombre-opción-usuario-constante-serie
)
▶▶

```

Descripción

nombre-autorización

Especifica el nombre de autorización bajo el cual un usuario o una aplicación se conecta a una base de datos federada. Este nombre se ha de correlacionar con un identificador bajo el cual se puede acceder a la fuente de datos indicada por *nombre-servidor*.

USER

El valor del registro especial USER. Cuando se especifica USER, el ID de autorización de la sentencia CREATE USER MAPPING se correlacionará con el ID de autorización de la fuente de datos que se especifica en la opción de usuario REMOTE_AUTHID.

SERVER *nombre-servidor*

Identifica la fuente de datos que se puede acceder bajo el ID de autorización de la correlación.

CREATE USER MAPPING

OPTIONS

Indica las opciones de usuario que se han de habilitar. Consulte el apartado “Opciones de usuario” en la página 1420 para ver las descripciones de *nombre-opción-usuario* y sus valores.

ADD

Habilita una o varias opciones de usuario.

nombre-opción-usuario

Nombra una opción de usuario que se utilizará para completar la correlación de usuarios que se está creando.

constante-serie

Especifica el valor para *nombre-opción-usuario* como una constante de serie de caracteres.

Notas

- No se puede crear una correlación de usuarios en una unidad de trabajo (UOW) determinada si la UOW ya incluye una sentencia SELECT que hace referencia a un apodo para una tabla o una vista de la fuente de datos que se ha de incluir en la correlación.

Ejemplos

Ejemplo 1: Para acceder a una fuente de datos llamada S1, necesita correlacionar el nombre de autorización y la contraseña para la base de datos local con el ID de usuario y la contraseña para S1. El nombre de autorización es RSPALTEN, y el ID de usuario y la contraseña que utiliza para S1 son SYSTEM y MANAGER, respectivamente.

```
CREATE USER MAPPING FOR RSPALTEN
SERVER S1
OPTIONS
( REMOTE_AUTHID 'SYSTEM',
  REMOTE_PASSWORD 'MANAGER' )
```

Ejemplo 2: Marc ya tiene acceso a la fuente de datos DB2. Ahora necesita acceder a una fuente de datos de Oracle, para poder crear uniones entre determinadas tablas DB2 y Oracle. Adquiere un nombre de usuario y una contraseña para la fuente de datos Oracle; el nombre de usuario es el mismo que su ID de autorización para la base de datos federada, pero las contraseñas de base de datos federada y Oracle son diferentes. Para poder acceder a Oracle desde la base de datos federada, debe correlacionar las dos contraseñas.

```
CREATE USER MAPPING FOR MARCR
SERVER ORACLE1
OPTIONS
( REMOTE_PASSWORD 'NZXCZY' )
```

CREATE VIEW

La sentencia CREATE VIEW crea una vista para una o más tablas, vistas o apodos.

Invocación

Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Es una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar dinámicamente (SQLSTATE 42509).

Autorización

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM, o bien
- Para cada tabla, vista o apodo identificado en cualquier selección completa:
 - Privilegio CONTROL para esa tabla o vista, o bien
 - Privilegio SELECT para esa tabla o vista

y como mínimo uno de los elementos siguientes:

- Autorización IMPLICIT_SCHEMA para la base de datos, si no existe el nombre de esquema, implícito o explícito, de la vista
- Privilegio CREATEIN para el esquema, si el nombre de esquema de la vista hace referencia a un esquema existente.

Si se crea una subvista, el ID de autorización de la sentencia debe:

- Ser el mismo que el definidor de la tabla raíz de la jerarquía de tablas.
- Tener SELECT WITH GRANT en la tabla principal de la subvista o el privilegio SELECT sobre la supervista no debe haberse otorgado a un usuario que no sea el definidor de la vista.

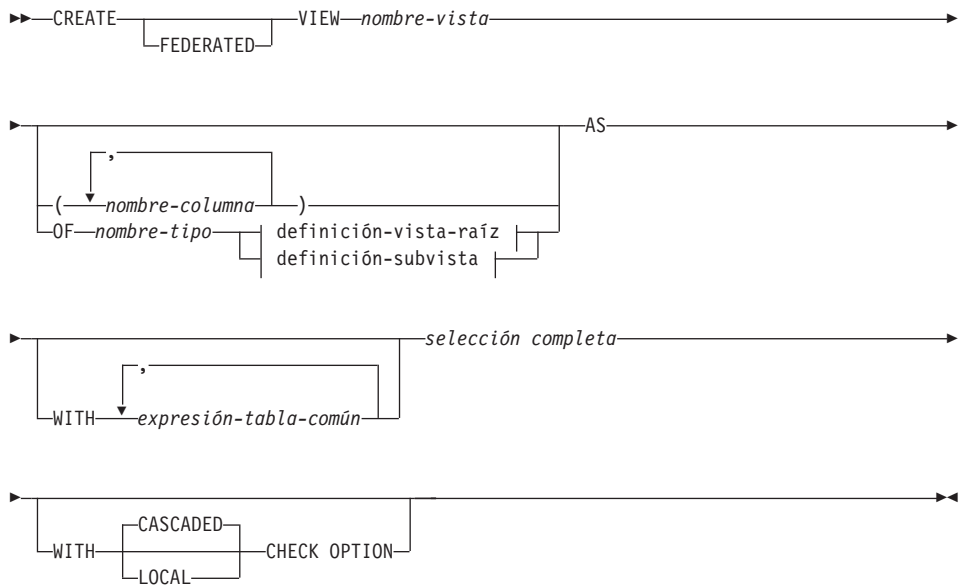
No se tienen en cuenta los privilegios de grupo para cualquier tabla o vista especificada en la sentencia CREATE VIEW.

Los privilegios no se tienen en cuenta al definir una vista en un apodo de base de datos federada. Los requisitos de autorización de la fuente de datos para la tabla o vista referenciada por el apodo se aplican cuando se procesa la consulta. El ID de autorización de la sentencia puede estar correlacionado con un ID de autorización remoto distinto.

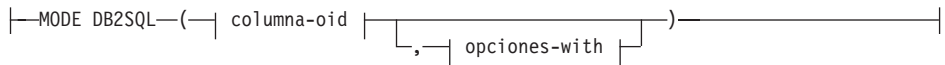
Si el definidor de una vista sólo puede crear la vista porque tiene autorización SYSADM, se le otorgará la autorización explícita DBADM con el propósito de crear la vista.

CREATE VIEW

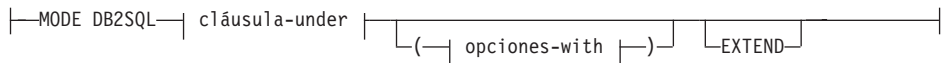
Sintaxis



definición-vista-raíz:



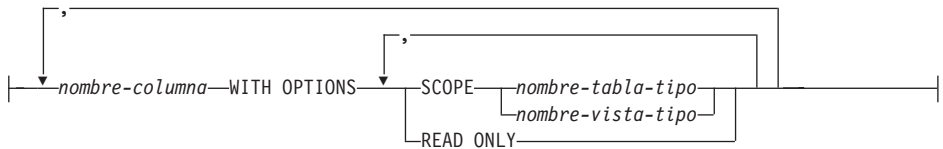
definición-subvista:



columna-oid:



opciones-with:

**cláusula-under:**

|—UNDER—*nombre-supervista*—INHERIT SELECT PRIVILEGES—|

Nota: En el “Capítulo 5. Consultas” en la página 471 se explica la sintaxis de *expresión-común-tabla* y *selección completa*.

Descripción**FEDERATED**

Indica que la vista que se crea hace referencia a un apodo o a una función de tabla OLEDB. Si la selección completa hace referencia, directa o indirectamente, a una función de tabla OLEDB o a un apodo, y no se especifica la palabra clave **FEDERATED**, se emite un aviso (SQLSTATE 01639) al someter la sentencia **CREATE VIEW**. Sin embargo, la vista todavía se crea.

Inversamente, si en la selección completa no se hace referencia, directa o indirectamente, a una tabla de función OLEDB ni a un apodo, y se especifica la palabra clave **FEDERATED**, se emite un error (SQLSTATE 429BA) al someter la sentencia **CREATE VIEW**. La vista no se creará.

nombre-vista

Indica el nombre de la vista. El nombre, incluido el calificador implícito o explícito, no debe identificar una tabla, vista, apodo o seudónimo descritos en el catálogo. El calificador no debe ser **SYSIBM**, **SYSCAT**, **SYSFUN** ni **SYSSTAT** (SQLSTATE 42939).

El nombre puede ser el mismo que el nombre de una vista no operativa (consulte el apartado “Vistas no operativas” en la página 967). En tal caso, la nueva vista especificada en la sentencia **CREATE VIEW** sustituirá la vista no operativa. El usuario recibirá un aviso (SQLSTATE 01595) cuando se sustituya una vista no operativa. No se devuelve ningún aviso si la aplicación se ha enlazado con la opción de enlace lógico **SQLWARN** establecida en **NO**.

nombre-columna

Indica los nombres de las columnas de la vista. Si se especifica una lista de nombres de columna, ésta debe constar de tantos nombres como columnas haya en la tabla resultante de la selección completa. Cada

CREATE VIEW

nombre-columna debe ser exclusivo y no calificado. Si no se especifica la lista de nombres de columnas, las columnas de la vista heredarán los nombres de las columnas de la tabla resultante de la selección completa.

Debe especificarse una lista de nombres de columna si la tabla resultante de la selección completa tiene nombres de columna duplicados o una columna sin nombre (SQLSTATE 42908). Una columna sin nombre es una columna derivada de una constante, función, expresión u operación de conjuntos que no se nombra utilizando la cláusula AS de la lista de selección.

OF *nombre-tipo*

Especifica que las columnas de la vista están basadas en los atributos del tipo estructurado identificado por *nombre-tipo*. Si se especifica *nombre-tipo* sin un nombre de esquema, el nombre de tipo se resuelve efectuando una búsqueda en los esquemas de la vía de acceso de SQL (definida por la opción de preproceso FUNCPATH en el caso del SQL estático y por el registro CURRENT PATH en el caso del SQL dinámico). El nombre de tipo debe ser el nombre de un tipo existente definido por el usuario (SQLSTATE 42704) y debe ser un tipo estructurado del que se pueda crear una instancia (SQLSTATE 428DP).

MODE DB2SQL

Esta cláusula se utiliza para especificar la modalidad de la vista con tipo. En este momento es la única modalidad válida a la que se da soporte.

UNDER *nombre-supervista*

Indica que la vista es una subvista de *nombre-supervista*. La supervista debe ser una vista existente (SQLSTATE 42704) y la vista debe estar definida mediante un tipo estructurado que sea el supertipo inmediato de *nombre-tipo* (SQLSTATE 428DB). El nombre de esquema de *nombre-vista* y *nombre-supervista* debe ser el mismo (SQLSTATE 428DQ). La vista identificada por *nombre-supervista* no debe tener ninguna subvista existente ya definida mediante *nombre-tipo* (SQLSTATE 42742).

Entre las columnas de la vista, se incluye la columna de identificador de objeto de la supervista con su tipo modificado para que sea REF(*nombre-tipo*), seguida de columnas basadas en los atributos de *nombre-tipo* (recuerde que el tipo incluye los atributos de su supertipo).

INHERIT SELECT PRIVILEGES

Cualquier usuario o grupo que sostenga un privilegio SELECT sobre la supervista recibirá un privilegio equivalente sobre la subvista recién creada. Se considera que el definidor de la subvista es el otorgante de este privilegio.

columna-OID

Define la columna de identificador de objeto para la vista con tipo.

REF IS *nombre-columna-OID* USER GENERATED

Especifica que en la vista se define una columna de identificador de objeto (OID) como la primera columna. Se necesita un OID para la vista raíz de una jerarquía de vistas (SQLSTATE 428DX). La vista debe ser una vista con tipo (debe estar presente la cláusula OF) que no sea una subvista (SQLSTATE 42613). El nombre de la columna se define como *nombre-columna-OID* y no puede ser el mismo que el nombre de cualquier atributo del tipo estructurado *nombre-tipo* (SQLSTATE 42711). La primera columna especificada en *selección completa* debe ser de tipo REF(*nombre-tipo*) (puede ser necesario que se convierta para que tenga el tipo adecuado). Si no se especifica UNCHECKED, la vista debe basarse en una columna sin posibilidad de nulos, en la que se asegura la unicidad mediante un índice (clave primaria, restricción de unicidad, índice de unicidad o columna OID). Esta columna vendrá referida como la *columna de identificador de objeto* o *columna de OID*. Las palabras clave USER GENERATED indican que el usuario debe proporcionar el valor inicial de la columna de OID cuando inserte una fila. Una vez que se haya insertado una fila, la columna de OID no podrá actualizarse (SQLSTATE 42808).

UNCHECKED

Define la columna identificadora de objeto de la definición de vista con tipo para asumir la unicidad aunque el sistema no pueda probar esta unicidad. Esto es así para utilizarlo con tablas o vistas que se están definiendo en un jerarquía de vistas con tipo donde el usuario sabe que los datos se ajustan a esta norma de unicidad pero no se ajustan a las normas que permiten al sistema probar esta unicidad. La opción UNCHECKED es obligatoria para jerarquías de vistas que comprenden varias jerarquías, o tablas o vistas preexistentes. Cuando se especifica UNCHECKED, el usuario debe asegurarse de que cada fila de la vista tiene un OID exclusivo. Si no se asegura esta propiedad y una vista contiene valores OID duplicados, puede producirse un error en una expresión de vía de acceso o en un operador Deref donde intervenga uno de los valores OID no exclusivos (SQLSTATE 21000).

opciones-with

Define opciones adicionales que se aplican a las columnas de una vista con tipo.

***nombre-columna* WITH OPTIONS**

Especifica el nombre de la columna para la que se especifican las opciones adicionales. El *nombre-columna* debe corresponderse con el nombre de un atributo definido en (no heredado por) el *nombre-tipo* de la vista. La columna debe ser un tipo de referencia (SQLSTATE 42842). No puede corresponderse con una columna que también exista en la

CREATE VIEW

supervista (SQLSTATE 428DJ). Sólo puede aparecer un nombre de columna en una cláusula WITH OPTIONS SCOPE de la sentencia (SQLSTATE 42613).

SCOPE

Identifica el ámbito de la columna de tipo de referencia. Debe especificarse un ámbito para cualquier columna que vaya a utilizarse como operando izquierdo de un operador de desreferencia o como argumento de la función Deref.

La especificación del ámbito de una columna de tipo de referencia puede diferirse a una sentencia ALTER VIEW subsiguiente (si no se hereda el ámbito) para permitir que se defina la vista o tabla de destino, normalmente en el caso de tablas y vistas que se hacen referencia mutuamente. Si no se especifica ningún ámbito para una columna de tipo de referencia de la vista y la columna de la vista o tabla subyacente tenía ámbito, la columna de tipo de referencia hereda el ámbito de la columna subyacente. La columna permanece sin ámbito si la columna de la vista o tabla subyacente no tenía ámbito. Consulte "Notas" en la página 966 para obtener más información sobre el ámbito y las columnas de tipo de referencia.

nombre-tabla-tipo

El nombre de una tabla con tipo. La tabla debe existir ya o ser la misma que el nombre de la tabla que se está creando (SQLSTATE 42704). El tipo de datos de *nombre-columna* debe ser REF(S), donde S es el tipo de *nombre-tabla-tipo* (SQLSTATE 428DM). No se realiza ninguna comprobación de los valores existentes de *nombre-columna* para asegurarse de si realmente los valores hacen referencia a filas existentes de *nombre-tabla-tipo*.

nombre-vista-tipo

El nombre de una vista con tipo. La vista debe existir ya o ser la misma que el nombre de la vista que se está creando (SQLSTATE 42704). El tipo de datos de *nombre-columna* debe ser REF(S), donde S es el tipo de *nombre-vista-tipo* (SQLSTATE 428DM). No se realiza ninguna comprobación de los valores existentes de *nombre-columna* para asegurarse de si realmente los valores hacen referencia a filas existentes de *nombre-vista-tipo*.

READ ONLY

Identifica la columna como columna de sólo lectura. Esta opción se utiliza para hacer que una columna sea de sólo lectura, de manera que las definiciones de las subvistas puedan especificar una expresión para la misma columna que sea de sólo lectura implícitamente.

AS

Identifica la definición de la vista.

WITH *expresión-tabla-común*

Define una expresión de tabla común para utilizarla con la selección completa que va a continuación. No puede especificarse una expresión de tabla común cuando se define una vista con tipo. Vea “expresión-común-tabla” en la página 519.

selección completa

Define la vista. En todo momento, la vista consta de las filas que se generarían si se ejecutara la sentencia SELECT. La selección completa no debe hacer referencia a variables del lenguaje principal, a marcadores de parámetros ni a tablas temporales declaradas. En cambio, una vista parametrizada se puede crear como función de tabla SQL. Vea “CREATE FUNCTION (SQL, escalar, de tabla o de fila)” en la página 761.

Para subvistas y vistas con tipo: La *selección completa* debe ajustarse a las normas siguientes, de lo contrario se devolverá un error (SQLSTATE 428EA a no ser que se especifique lo contrario).

- La selección completa no debe incluir referencias a las funciones NODENUMBER o PARTITION, a las funciones no deterministas o a las funciones definidas para tener acción externa.
- El cuerpo de la vista debe consistir en una sola subselección o en UNION ALL de dos o mas subselecciones. Supongamos que cada una de las subselecciones que participan directamente en el cuerpo de la vista se llama una *rama* de la vista. Una vista puede tener una o más ramas.
- La cláusula FROM de cada rama debe consistir de una sola tabla o vista (no necesariamente de tipo), llamada la tabla o vista *subyacente* de esa rama.
- La tabla o vista subyacente de cada rama debe estar en una jerarquía separada (por ejemplo, una vista puede que no tenga ramas múltiples con sus tablas o vistas subyacentes en la misma jerarquía).
- Ninguna de las ramas de una definición de vista con tipo puede que especifique GROUP BY o HAVING.
- Si el cuerpo de la vista contiene UNION ALL, entonces la vista raíz de la jerarquía debe especificar la opción UNCHECKED para su columna OID.

Para una jerarquía de vistas y subvistas: BR1 y BR2 son las ramas que aparecen en las definiciones de vistas en la jerarquía. T1 debe ser la tabla subyacente o vista de BR1 y T2 debe ser la tabla o vista subyacente de BR2. Entonces:

- Si T1 y T2 no están en la misma jerarquía, entonces la vista raíz de la jerarquía de vistas debe especificar la opción UNCHECKED para su columna OID.

CREATE VIEW

- Si T1 y T2 están en la misma jerarquía, entonces BR1 y BR2 deben contener predicados o cláusulas ONLY que sean suficientes para garantizar que sus conjuntos de filas no están unidos.

Para subvistas con tipo definidas utilizando EXTEND AS: Para cada rama en el cuerpo de la subvista:

- La tabla subyacente de cada rama debe ser una subtabla (no necesariamente la correspondiente) de alguna tabla subyacente de la supervista inmediata.
- Las expresiones de la lista SELECT deben poder asignarse a las columnas no heredadas de la subvista (SQLSTATE 42854).

Para subvistas con tipo definidas utilizando AS sin EXTEND:

- Para cada rama del cuerpo de la subvista, las expresiones en la lista SELECT deben poder asignarse a los tipos declarados de las columnas heredadas y no heredadas de la subvista (SQLSTATE 42854).
- La expresión OID de cada rama de una jerarquía de la subvista debe ser equivalente (excepto para la conversión del tipo de datos) a la expresión OID en la rama en la misma jerarquía en la vista raíz.
- La expresión para una columna no definida (implícita o explícitamente) como READ ONLY en una supervista debe ser equivalente a todas las ramas en la misma jerarquía subyacente en sus subvistas.

WITH CHECK OPTION

Especifica la restricción según la cual cada fila que se haya insertado o actualizado a través de la vista debe ajustarse a la definición de dicha vista. Una fila que no se ajusta a la definición de la vista es una fila que no cumple las condiciones de búsqueda de la vista.

WITH CHECK OPTION no debe especificarse si la vista es de sólo lectura (SQLSTATE 42813). Si se especifica WITH CHECK OPTION para una vista actualizable que no permite inserciones, la restricción se aplicará solamente a las actualizaciones.

No debe especificarse WITH CHECK OPTION si la vista hace referencia a la función NODENUMBER o PARTITION, a una función no determinista o a una función con acción externa (SQLSTATE 42997).

No debe especificarse WITH CHECK OPTION si la vista es una vista con tipo (SQLSTATE 42997).

No debe especificarse WITH CHECK OPTION si un apodo es el destino de actualización de la vista.

Si se omite WITH CHECK OPTION, la definición de la vista no se utilizará en la comprobación de ninguna operación de inserción o de actualización que utilicen esa vista. Si la vista depende directa o indirectamente de otra vista que incluya WITH CHECK OPTION, durante

las operaciones de inserción y actualización es posible que se siga produciendo algún tipo de comprobación. Dado que no se utiliza la definición de la vista, se podrían insertar o actualizar filas a través de la vista que no se ajustasen a la definición de la vista.

CASCADED

La restricción **WITH CASCADED CHECK OPTION** en una vista *V* significa que *V* hereda las condiciones de búsqueda como restricciones de cualquier vista actualizable de la que *V* depende. Además, todas las vistas actualizables que dependen de *V* también están sometidas a estas restricciones. De esta forma, las condiciones de búsqueda de *V* y todas las vistas de las que *V* depende se unen mediante **AND** para formar una restricción que se aplica a las inserciones o actualizaciones de *V* o de cualquier vista dependiente de *V*.

LOCAL

La restricción **WITH LOCAL CHECK OPTION** en una vista *V* significa que la condición de búsqueda de *V* se aplica como restricción a las inserciones o actualizaciones de *V* o de cualquier vista que sea dependiente de *V*.

La diferencia entre **CASCADED** y **LOCAL** se explica en el ejemplo siguiente. Tome en consideración las siguientes vistas actualizables (sustituyendo *Y* en las cabeceras de columna de la tabla que sigue):

- V1 definida en la tabla T
- V2 definida en V1 **WITH Y CHECK OPTION**
- V3 definida en V2
- V4 definida en V3 **WITH Y CHECK OPTION**
- V5 definida en V4

La tabla siguiente muestra las condiciones de búsqueda con las que se comparan las filas insertadas o actualizadas:

	Y es LOCAL	Y es CASCADED
V1 se comprueba con:	ninguna vista	ninguna vista
V2 se comprueba con:	V2	V2, V1
V3 se comprueba con:	V2	V2, V1
V4 se comprueba con:	V2, V4	V4, V3, V2, V1
V5 se comprueba con:	V2, V4	V4, V3, V2, V1

Tome en consideración la siguiente vista actualizable que muestra el efecto de **WITH CHECK OPTION** utilizando la opción **CASCADED** por omisión:

```
CREATE VIEW V1 AS SELECT COL1 FROM T1 WHERE COL1 > 10

CREATE VIEW V2 AS SELECT COL1 FROM V1 WITH CHECK OPTION

CREATE VIEW V3 AS SELECT COL1 FROM V2 WHERE COL1 < 100
```

CREATE VIEW

La siguiente sentencia INSERT que utiliza *V1* será satisfactoria porque *V1* no tiene WITH CHECK OPTION ni tampoco depende de ninguna otra vista que lo tenga.

```
INSERT INTO V1 VALUES(5)
```

La siguiente sentencia INSERT que utiliza *V2* dará lugar a error porque *V2* tiene WITH CHECK OPTION y la inserción generaría una fila que no se ajustaría con la definición de *V2*.

```
INSERT INTO V2 VALUES(5)
```

La siguiente sentencia INSERT que utiliza *V3* dará lugar a un error aun no teniendo WITH CHECK OPTION porque *V3* depende de *V2*, que sí tiene especificado WITH CHECK OPTION (SQLSTATE 44000).

```
INSERT INTO V3 VALUES(5)
```

La siguiente sentencia INSERT que utiliza *V3* será satisfactoria aunque no se ajuste a la definición de *V3* (*V3* no tiene WITH CHECK OPTION); se ajusta a la definición de *V2*, que sí tiene WITH CHECK OPTION.

```
INSERT INTO V3 VALUES(200)
```

Notas

- La creación de una vista con un nombre de esquema que no exista todavía dará como resultado la creación implícita de dicho esquema siempre que el ID de autorización de la sentencia tenga la autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN en el esquema se otorga a PUBLIC.
- Las columnas de la vista heredan el atributo NOT NULL WITH DEFAULT de la tabla o vista base excepto cuando las columnas derivan de una expresión. Al insertar o actualizar una fila en una vista actualizable, se comprueba comparándola con las restricciones (clave primaria, integridad de referencia y control) si es que hay alguna definida en la tabla base.
- No se puede crear una nueva vista si ésta utiliza en su definición una vista no operativa. (SQLSTATE 51024).
- Esta sentencia no permite utilizar tablas temporales declaradas (SQLSTATE 42995).
- **Vistas suprimibles:** Una vista es *suprimible* si se cumplen todas las condiciones siguientes:
 - cada cláusula FROM de la selección completa exterior identifica una sola tabla base (sin cláusula OUTER), una vista suprimible (sin cláusula OUTER), una expresión de tabla anidada suprimible o una expresión de tabla común suprimible (no puede identificar un apodo)
 - la selección completa exterior no incluye una cláusula VALUES
 - la selección completa exterior no incluye una cláusula GROUP BY ni una cláusula HAVING

- la selección completa exterior no incluye funciones de columna en la lista de selección
- la selección completa exterior no incluye operaciones SET (UNION, EXCEPT o INTERSECT) a excepción de UNION ALL
- las tablas base en los operandos de una UNION ALL no deben ser iguales y cada operando debe poderse suprimir
- la lista de selección de la selección completa exterior no incluye DISTINCT
- **Vistas actualizables:** Una columna de una vista es *actualizable* si se cumplen todas las condiciones siguientes:
 - la vista es suprimible
 - la resolución de la columna devuelve una columna de una tabla base (sin utilizar una operación de desreferencia) y no se especifica la opción READ ONLY
 - todas las columnas correspondientes de los operandos de una UNION ALL tienen tipos de datos que coinciden exactamente (incluyendo longitud o precisión y escala) y valores por omisión que coinciden, si la selección completa de la vista incluye una UNION ALL

Una vista es *actualizable* si CUALQUIER columna de la vista es actualizable.

- **Vistas insertables:** Una vista es *insertable* si TODAS las columnas de la vista son actualizables y la selección completa de la vista no incluye UNION ALL.
- **Vistas de sólo lectura:** Una vista es de *sólo lectura* si NO es suprimible. La columna READONLY de la vista de catálogo SYSCAT.VIEWS indica si una vista es de sólo lectura.
- Las expresiones de tabla comunes y las expresiones de tabla anidadas siguen el mismo conjunto de reglas para determinar si son suprimibles, actualizables, insertables o de sólo lectura.
- **Vistas no operativas:** Una *vista no operativa* es una vista que ya no está disponible para las sentencias de SQL. Una vista deja de ser operativa si:
 - Se revoca un privilegio del que depende la definición de dicha vista.
 - Se elimina un objeto, como una tabla, un apodo, un seudónimo o una función, del que depende la definición de dicha vista.
 - Otra vista, de la cual depende la definición de la vista, deja de ser operativa.
 - Una vista que es la supervista de la definición de vista (la subvista) se vuelve no operativa.

En otras palabras, una vista no operativa es aquélla en la que se ha eliminado involuntariamente la definición de vista. Por ejemplo, al eliminar un seudónimo, cualquier vista definida con ese seudónimo deja de ser

CREATE VIEW

operativa. También se considerarán no operativas todas las vistas dependientes, y los paquetes que dependen de la misma ya no se consideran válidos.

Hasta que no se vuelva a crear o eliminar explícitamente una vista no operativa, no podrán compilarse las sentencias que utilicen esa vista (SQLSTATE 51024), exceptuando las sentencias CREATE ALIAS, CREATE VIEW, DROP VIEW y COMMENT ON TABLE. Hasta que no se haya eliminado explícitamente la vista no operativa, no se puede utilizar su nombre calificado para crear otra tabla o seudónimo (SQLSTATE 42710).

Una vista no operativa puede volver a crearse emitiendo una sentencia CREATE VIEW mediante el texto de definición de la vista no operativa. Este texto de definición de vista se almacena en la columna TEXT del catálogo SYSCAT.VIEWS. Cuando se vuelve a crear una vista no operativa, es necesario otorgar de forma explícita todos los privilegios que otros necesitan en dicha vista, debido al hecho de que se suprimen todos los registros de autorizaciones en una vista si la vista se marca como no operativa. Observe que no es necesario eliminar de manera explícita la vista no operativa para volverla a crear. La emisión de una sentencia CREATE VIEW que utilice el mismo *nombre-vista* que una vista no operativa hará que se sustituya la vista no operativa, y la sentencia CREATE VIEW emitirá un aviso (SQLSTATE 01595).

Las vistas no operativas se indican mediante una X en la columna VALID de la vista de catálogo SYSCAT.VIEWS y una X en la columna STATUS de la vista de catálogo SYSCAT.TABLES.

- **Privilegios**

El definidor de una vista siempre recibe el privilegio SELECT para la vista, así como el derecho a eliminar la vista. El definidor de una vista obtendrá el privilegio CONTROL para la vista sólo si tiene el privilegio CONTROL para todas las tablas base, vistas o apodos identificados en la selección completa, o si tiene autorización SYSADM o DBADM.

El definidor de la vista recibe los privilegios INSERT, UPDATE, UPDATE a nivel de columna o DELETE para la vista si ésta no es de sólo lectura y el definidor tiene los privilegios correspondientes para los objetos subyacentes.

El definidor de una vista sólo adquiere privilegios si los privilegios de los cuales aquéllos derivan ya existen cuando se crea la vista. El definidor debe tener estos privilegios directamente o porque PUBLIC tiene el privilegio. Los privilegios no se tienen en cuenta al definir una vista para un apodo de servidor federado. Sin embargo, al utilizar una vista en un apodo, el ID de autorización de usuario debe tener los privilegios de selección válidos en la tabla o vista a la que el apodo hace referencia en la fuente de datos. De lo contrario, se emite un error. No se tienen en cuenta los privilegios pertenecientes a grupos de los que forma parte el definidor.

Cuando se crea una subvista, se otorgan automáticamente sobre la misma los privilegios SELECT sostenidos sobre la supervista inmediata.

- **Columnas REF y de ámbito**

Cuando seleccione una columna de tipo de referencia en la selección completa de una definición de vista, tenga en cuenta el tipo de destino y el ámbito que sean necesarios.

- Si el tipo de destino y el ámbito necesarios son los mismos que los de la vista o tabla subyacente, ya puede seleccionar la columna.
- Si se debe cambiar el ámbito, utilice la cláusula WITH OPTIONS SCOPE para definir la vista o tabla con ámbito necesarias.
- Si se debe cambiar el tipo de destino de la referencia, se debe convertir la columna primero en el tipo de representación de la referencia y después en el tipo de referencia nuevo. En este caso, el ámbito se puede especificar en la conversión hacia el tipo de referencia o mediante la cláusula WITH OPTIONS SCOPE. Por ejemplo, suponga que selecciona la columna Y definida como REF(TYP1) SCOPE TAB1. Desea que se defina como REF(VTYP1) SCOPE VIEW1. El elemento de la lista de selección sería el siguiente:

```
CAST(CAST(Y AS VARCHAR(16) FOR BIT DATA) AS REF(VTYP1) SCOPE VIEW1)
```

- **Columnas de identidad** Se considera que una columna de una vista es una columna de identidad si el elemento de la columna correspondiente en la selección completa de la definición de vista es el nombre de una columna de identidad de una tabla, o el nombre de una columna de una vista que, directa o indirectamente, se correlaciona con el nombre de una columna de identidad de una tabla base.

En todos los demás casos, las columnas de una vista no obtendrán el atributo de identidad. Por ejemplo:

- La lista de selección de la definición de vista contiene varias instancias del nombre de una columna de identidad (es decir, se selecciona la misma columna más de una vez)
- la definición de la vista incluye una operación de unión
- una columna de la definición de la vista incluye una expresión que hace referencia a una columna de identidad
- la definición de la vista incluye una operación UNION

Cuando se inserta en una vista para la que la lista de selección de la definición de vista incluye, directa o indirectamente, el nombre de una columna de identidad de una tabla base, son aplicables las mismas reglas que si la sentencia INSERT hiciera referencia directamente a la columna de identidad de la tabla base.

- **Vistas federadas** Una vista federada es una vista que incluye una referencia a un apodo situado en algún lugar de la selección completa. La presencia de este apodo cambia el modelo de autorización utilizado para la vista

CREATE VIEW

durante la creación y cuando la vista se utiliza posteriormente en una consulta. Si se crea una vista que hace referencia a un apodo y no se incluye la palabra clave `FEDERATED`, se emite un aviso para indicar que los requisitos de autorización para la vista son diferentes debido a la referencia a un apodo.

Un apodo no tiene privilegios DML asociados y, por lo tanto, cuando se crea la vista, no se realiza ninguna comprobación de privilegios para determinar si el definidor de la vista tiene acceso al apodo o a la vista o tabla de fuente de datos subyacente. El control de privilegios para referencias a tablas o vistas de la base de datos federada se lleva a cabo como siempre, siendo necesario que el definidor de la vista tenga al menos el privilegio `SELECT` sobre esos objetos.

Cuando subsiguientemente se hace referencia a una vista federada en una consulta, los apodos que resultan de consultas en la fuente de datos e ID de autorización que emite la consulta (o el ID de autorización remoto al que corresponde) deben tener los privilegios necesarios para tener acceso a la vista o tabla de fuente de datos. El ID de autorización que emite la consulta que hace referencia a la vista federada no es obligatorio que tenga ningún privilegio adicional en vista o tablas (no federadas) que existen en el servidor federado.

Ejemplos

Ejemplo 1: Cree una vista denominada `MA_PROJ` en la tabla `PROJECT` que sólo contenga las filas con un número de proyecto (`PROJNO`) que empiece por las letras 'MA'.

```
CREATE VIEW MA_PROJ AS SELECT *
FROM PROJECT
WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

Ejemplo 2: Cree una vista como la del ejemplo 1, pero seleccione sólo las columnas para el número de proyecto (`PROJNO`), nombre de proyecto (`PROJNAME`) y empleado encargado del proyecto (`RESPEMP`).

```
CREATE VIEW MA_PROJ
AS SELECT PROJNO, PROJNAME, RESPEMP
FROM PROJECT
WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

Ejemplo 3: Cree una vista como la del ejemplo 2, pero en la vista, llame a la columna para el empleado encargado del proyecto `IN_CHARGE`.

```
CREATE VIEW MA_PROJ
(PROJNO, PROJNAME, IN_CHARGE)
AS SELECT PROJNO, PROJNAME, RESPEMP
FROM PROJECT
WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```


Nota: Aunque sólo se cambie uno de los nombres de columna, los nombres de las tres columnas de la vista deben listarse entre los paréntesis que siguen a MA_PROJ.

Ejemplo 4: Cree una vista llamada PRJ_LEADER que contenga las cuatro primeras columnas (PROJNO, PROJNAME, DEPTNO, RESPEMP) de la tabla PROJECT junto con el apellido (LASTNAME) de la persona que es responsable del proyecto (RESPEMP). Obtendremos el nombre de la tabla EMPLOYEE emparejando EMPNO de EMPLOYEE con RESPEMP de PROYECT.

```
CREATE VIEW PRJ_LEADER
AS SELECT PROJNO, PROJNAME, DEPTNO, RESPEMP, LASTNAME
FROM PROJECT, EMPLOYEE
WHERE RESPEMP = EMPNO
```

Ejemplo 5: Cree una vista como en el ejemplo 4, pero que además de mostrar las columnas PROJNO, PROJNAME, DEPTNO, RESPEMP y LASTNAME, que también muestre la paga total (SALARY + BONUS + COMM) del empleado responsable. Asimismo, seleccione sólo aquellos proyectos cuyo empleo de personal principal (PRSTAFF) sea mayor que 1.

```
CREATE VIEW PRJ_LEADER
(PROJNO, PROJNAME, DEPTNO, RESPEMP, LASTNAME, TOTAL_PAY )
AS SELECT PROJNO, PROJNAME, DEPTNO, RESPEMP, LASTNAME, SALARY+BONUS+COMM
FROM PROJECT, EMPLOYEE
WHERE RESPEMP = EMPNO
AND PRSTAFF > 1
```

La especificación de la lista de nombres de columnas se puede evitar unificando la expresión SALARY+BONUS+COMM con el nombre TOTAL_PAY en la selección completa.

```
CREATE VIEW PRJ_LEADER
AS SELECT PROJNO, PROJNAME, DEPTNO, RESPEMP,
LASTNAME, SALARY+BONUS+COMM AS TOTAL_PAY
FROM PROJECT, EMPLOYEE
WHERE RESPEMP = EMPNO AND PRSTAFF > 1
```

Ejemplo 6: Dado el conjunto de tablas y vistas mostrado en el diagrama siguiente:

CREATE VIEW

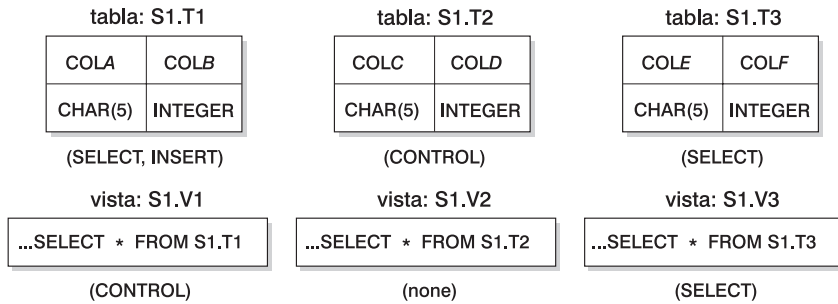


Figura 14. Tablas y vistas para el ejemplo 6

Se han otorgado al usuario ZORPIE (que no tiene la autorización DBADM ni SYSADM) los privilegios que se muestran entre corchetes debajo de cada objeto:

1. ZORPIE obtendrá el privilegio CONTROL en la vista que cree con:

```
CREATE VIEW VA AS SELECT * FROM S1.V1
```

porque tiene CONTROL en S1.V1.⁸⁷ No importa qué privilegios tenga, si tiene alguno, en la tabla base principal.

2. ZORPIE no estará autorizada a crear la vista:

```
CREATE VIEW VB AS SELECT * FROM S1.V2
```

como ZORPIE no tiene ni CONTROL ni SELECT en S1.V2. No importa el hecho de que tenga CONTROL en la tabla base principal (S1.T2).

3. ZORPIE obtendrá el privilegio CONTROL en la vista que cree con:

```
CREATE VIEW VC (COLA, COLB, COLC, COLD)  
AS SELECT * FROM S1.V1, S1.T2  
WHERE COLA = COLC
```

porque la selección completa de ZORPIE.VC hace referencia a la vista S1.V1 y a la tabla S1.T2 y dispone de CONTROL en ambas. Observe que la vista VC es de sólo lectura, por lo tanto ZORPIE no obtiene los privilegios INSERT, UPDATE ni DELETE.

4. ZORPIE obtendrá el privilegio SELECT en la vista que cree con:

```
CREATE VIEW VD (COLA, COLB, COLE, COLF)  
AS SELECT * FROM S1.V1, S1.V3  
WHERE COLA = COLE
```

87. CONTROL en S1.V1 debe habérselo otorgado a ZORPIE alguien con autorización DBADM o SYSADM.

porque la selección completa de ZORPIE.VD hace referencia a las dos vistas S1.V1 y S1.V3, en una de las cuales sólo tiene el privilegio SELECT, y en la otra tiene el privilegio CONTROL. Se le otorga el menor de los dos privilegios, SELECT, en ZORPIE.VD.

5. ZORPIE obtendrá el privilegio INSERT, UPDATE y DELETE con GRANT OPTION y el privilegio SELECT en la vista VE en la siguiente definición de vista.

```
CREATE VIEW VE  
AS SELECT * FROM S1.V1  
WHERE COLA > ANY  
    (SELECT COLE FROM S1.V3)
```

Los privilegios de ZORPIE en VE se determinan principalmente de acuerdo con sus privilegios en S1.V1. Como sólo se hace referencia a S1.V3 en una subconsulta, sólo necesita el privilegio SELECT en S1.V3 para crear la vista VE. La persona que define la vista sólo obtiene CONTROL en la vista si tiene CONTROL en todos los objetos a los que se hace referencia en la definición de vista. ZORPIE no tiene CONTROL en S1.V3, en consecuencia no obtiene CONTROL en VE.

CREATE WRAPPER

CREATE WRAPPER

La sentencia CREATE WRAPPER registra un reiniciador—mecanismo por el cual un servidor federado puede interactuar con una categoría determinada de fuentes de datos—en una base de datos federada.

Invocación

Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

Autorización

El ID de autorización de la sentencia debe tener la autorización SYSADM o DBADM.

Sintaxis

```
►► CREATE WRAPPER—nombre-reiniciador—┬──────────────────────────────────┬──────────────────►►  
                                          └──LIBRARY—'nombre-biblioteca'──┘
```

Descripción

nombre-reiniciador

Nombra el reiniciador. Puede ser:

- Un nombre predefinido. Si se especifica un nombre predefinido, el servidor federado asigna automáticamente un valor por omisión a '*nombre-biblioteca*'.

Los nombres predefinidos son:

DRDA	Para todas las fuentes de datos de la familia DB2
NET8	Para las fuentes de datos Oracle que el software de cliente Net8 de Oracle soporta
OLEDB	Para todos los proveedores de OLE DB soportados por Microsoft OLE DB
SQLNET	Para todas las fuentes de datos Oracle que el software de cliente SQL*Net de Oracle soporta

- Un nombre suministrado por el usuario. Si se proporciona dicho nombre, también es necesario especificar '*nombre-biblioteca*'.

LIBRARY '*nombre-biblioteca*'

Nombra el archivo que contiene el módulo de reiniciador. La opción LIBRARY sólo es necesaria si se utiliza un *nombre-reiniciador* proporcionado por el usuario. Esta opción no debe utilizarse cuando se proporciona un *nombre-reiniciador* predefinido. Los nombres de archivo por

omisión correspondientes a los *nombres-reiniciador* predefinidos son:

Tabla 28. Nombres de archivo por omisión para la opción *LIBRARY*

Plataforma	DRDA	SQLNET	NET8	OLEDB
AIX	libdrda.a	libsqlnet.a	libnet8.a	–
HP-UX	libdrda.sl	libsqlnet.sl	libnet8.sl	–
Linux	libdrda.so	libsqlnet.so	–	–
SOLARIS	libdrda.so	libsqlnet.so	libnet8.so	–
WINNT	drda.dll	sqlnet.dll	net8.dll	db2oledb.dll

Notas

Consulte *Suplemento de instalación y configuración* para obtener más información sobre cómo seleccionar y definir reiniciador.

Ejemplos

Ejemplo 1: Registre un reiniciador que el servidor federado pueda utilizar para interactuar con una fuente de datos Oracle que el software de cliente SQL*Net de Oracle soporta. Utilice el nombre predefinido.

```
CREATE WRAPPER SQLNET
```

Ejemplo 2: Registre un reiniciador que el servidor federado de un sistema AIX pueda utilizar para interactuar con fuentes de datos DB2 para VM y VSE. Especifique un nombre para indicar que estas fuentes de datos se utilizan para prueba.

```
CREATE WRAPPER TEST
LIBRARY 'libsqlds.a'
```

La extensión del nombre de biblioteca (a) indica que el reiniciador TEST es para las fuentes de datos que residen en un sistema AIX.

DECLARE CURSOR

DECLARE CURSOR

La sentencia DECLARE define un cursor.

Invocación

Aunque un recurso SQL interactivo pueda proporcionar una interfaz que dé la apariencia de ejecución interactiva, esta sentencia sólo puede incluirse dentro de un programa de aplicación. No se trata de una sentencia ejecutable y no puede prepararse dinámicamente.

Autorización

El término “sentencia SELECT del cursor” se utiliza para especificar las reglas de autorización. La sentencia SELECT del cursor es una de las siguientes:

- La sentencia-select preparada que se identifica por el *nombre-sentencia*
- La *sentencia-select* especificada.

Por cada tabla o vista que esté identificada (ya sea directamente o mediante un seudónimo) en la sentencia SELECT del cursor, el ID de autorización de la sentencia debe tener como mínimo uno de los privilegios siguientes:

- Autorización SYSADM o DBADM.
- Para cada tabla o vista identificada en la *sentencia-select*:
 - Privilegio SELECT para la tabla o vista, o bien
 - Privilegio CONTROL para la tabla o vista.

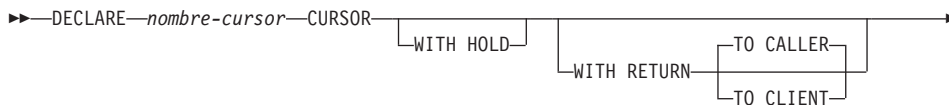
Si se especifica *nombre-sentencia*:

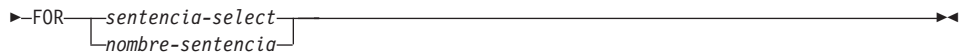
- El ID de autorización de la sentencia es el ID de autorización de ejecución.
- El control de autorizaciones se realiza al preparar la sentencia-select.
- El cursor no puede abrirse si la sentencia-select no está preparada correctamente.

Si se especifica *sentencia-select*:

- No se comprueban los privilegios GROUP.
- El ID de autorización de la sentencia es el ID de autorización especificado durante la preparación del programa.

Sintaxis





Descripción

nombre-cursor

Especifica el nombre del cursor que se ha creado al ejecutar el programa fuente. Este nombre no debe ser el mismo que el de ningún otro cursor que esté declarado en el programa fuente. Antes de utilizar el cursor es preciso abrirlo (véase el apartado “OPEN” en la página 1094).

WITH HOLD

Mantiene recursos en varias unidades de trabajo. El efecto del atributo del cursor WITH HOLD es el siguiente:

- En las unidades de trabajo que finalizan con COMMIT:
 - Los cursores abiertos definidos con WITH HOLD permanecen abiertos. El cursor se sitúa antes de la siguiente fila lógica de la tabla resultante.

Si se emite la sentencia DISCONNECT después de la sentencia COMMIT para una conexión con cursores WITH HOLD, los cursores mantenidos deben cerrarse explícitamente, porque si no se supondrá que la conexión ha realizado cierto trabajo (simplemente por tener abiertos cursores WITH HOLD aun sin haber emitido sentencias de SQL) y fallará la sentencia DISCONNECT.
 - Se liberan todos los bloqueos, excepto los bloqueos que protegen la posición actual del cursor, de los cursores WITH HOLD abiertos. Los bloqueos mantenidos incluyen los bloqueos sobre la tabla y para los entornos paralelos, los bloqueos sobre las filas en las que los cursores están situados actualmente. Los bloqueos sobre paquetes y secciones SQL dinámicas (si las hay) se mantienen.
 - Las operaciones válidas en los cursores definidos WITH HOLD que están inmediatamente a continuación de una petición COMMIT son:
 - FETCH: Lee la siguiente fila del cursor.
 - CLOSE: Cierra el cursor.
 - UPDATE y DELETE CURRENT OF CURSOR sólo son válidas para aquellas filas que se recuperan dentro de la misma unidad de trabajo.
 - Se liberan los localizadores de LOB.
- Para las unidades de trabajo que finalizan con ROLLBACK:
 - Se cierran todos los cursores abiertos
 - Se liberan todos los bloqueos adquiridos durante la unidad de trabajo
 - Se liberan los localizadores de LOB.

DECLARE CURSOR

- Para el caso especial de COMMIT:
 - Los paquetes pueden volver a crearse ya sea explícitamente, enlazándolos, o implícitamente, debido a que fueron invalidados y luego vueltos a crear dinámicamente la primera vez que se ha hecho referencia a los mismos. Todos los cursores mantenidos se cierran cuando se vuelve a enlazar el paquete. Ello puede provocar errores en una posterior ejecución.

WITH RETURN

Esta cláusula indica que el cursor está pensado para ser utilizado como conjunto resultante de un procedimiento almacenado. WITH RETURN sólo es aplicable si el código fuente de un procedimiento almacenado contiene la sentencia DECLARE CURSOR. En los demás casos, el precompilador puede aceptar la cláusula, pero no tiene ningún efecto.

Dentro de un procedimiento SQL, los cursores declarados mediante la cláusula WITH RETURN que todavía están abiertos cuando finaliza el procedimiento SQL, definen los conjuntos resultantes del procedimiento SQL. Todos los demás cursores abiertos de un procedimiento SQL se cierran cuando finaliza el procedimiento SQL. Dentro de un procedimiento almacenado externo (un procedimiento no definido utilizando LANGUAGE SQL), la cláusula WITH RETURN no tiene ningún efecto, y todos los cursores que están abiertos cuando finaliza un procedimiento externo se consideran que son los conjuntos resultantes.

TO CALLER

Especifica que el cursor puede devolver un conjunto resultante al llamador. Por ejemplo, si el llamador es otro procedimiento almacenado, el conjunto resultante se devuelve a ese procedimiento almacenado. Si el llamador es una aplicación cliente, el conjunto resultante se devuelve a la aplicación cliente.

TO CLIENT

Especifica que el cursor puede devolver un conjunto resultante a la aplicación cliente. Este cursor es invisible para cualquier procedimiento anidado intermedio.

sentencia-select

Identifica la sentencia SELECT del cursor. La *sentencia-select* no debe incluir marcadores de parámetros, pero puede incluir referencias a variables del lenguaje principal. Las declaraciones de variables del lenguaje principal deben preceder a la sentencia DECLARE CURSOR en el programa fuente. En el apartado “sentencia-select” en la página 518 se explica la *sentencia de selección*.

nombre-sentencia

La sentencia SELECT del cursor es la sentencia SELECT preparada que se indica por el *nombre-sentencia* cuando está abierto el cursor. El

nombre-sentencia no debe ser igual a ningún otro *nombre-sentencia* que esté especificado en otra sentencia DECLARE CURSOR del programa fuente.

En el apartado “PREPARE” en la página 1099 hallará una explicación de las sentencias SELECT preparadas.

Notas

- Un programa invocado desde otro programa o desde otro archivo fuente dentro del mismo programa no puede utilizar el cursor que fue abierto por el programa llamador.
- Los procedimientos almacenados no anidados, no definidos con LANGUAGE SQL, utilizan por omisión la opción WITH RETURN TO CALLER si DECLARE CURSOR está especificado sin una cláusula WITH RETURN, y el cursor se deja abierto en el procedimiento. Esto proporciona compatibilidad con procedimientos almacenados pertenecientes a versiones anteriores, en las cuales los procedimientos almacenados pueden devolver conjuntos resultantes a las aplicaciones cliente correspondientes. Para evitar este comportamiento, cierre todos los cursores abiertos en el procedimiento.
- Si la sentencia SELECT de un cursor contiene CURRENT DATE, CURRENT TIME o CURRENT TIMESTAMP, todas las referencias a estos registros especiales producirán el mismo valor en cada FETCH. Este valor se determina al abrir el cursor.
- Para lograr un proceso de datos más eficaz, el gestor de bases de datos puede agrupar los datos en bloques para cursores de sólo lectura cuando se recuperan datos de un servidor remoto. Mediante la cláusula FOR UPDATE, el gestor de bases de datos puede decidir si un cursor es actualizable o no. La posibilidad de actualización también se utiliza para determinar la selección de la vía de acceso. Si un cursor no se va a utilizar en una sentencia UPDATE con posición o DELETE, debería declararse como FOR READ ONLY.
- Un cursor en estado abierto designa una tabla de resultado y una posición relativa para las filas de dicha tabla. La tabla es la tabla resultante especificada por la sentencia SELECT del cursor.
- Un cursor es *suprimible* si se cumplen todas las condiciones siguientes:
 - cada cláusula FROM de la selección completa externa identifica una sola tabla base o vista suprimible (no puede identificar una expresión de tabla común, una expresión de tabla anidada ni un apodo) sin utilizar la cláusula OUTER
 - la selección completa externa no incluye una cláusula VALUES
 - la selección completa externa no incluye una cláusula GROUP BY ni una cláusula HAVING
 - la selección completa externa no incluye funciones de columna en la lista de selección

DECLARE CURSOR

- la selección completa externa no incluye operaciones SET (UNION, EXCEPT o INTERSECT) a excepción de UNION ALL
- la lista de selección de la selección completa externa no incluye DISTINCT
- la sentencia-select no incluye una cláusula ORDER BY
- la sentencia-select no incluye una cláusula FOR READ ONLY⁸⁸
- se cumple una o más de las condiciones siguientes:
 - la cláusula FOR UPDATE⁸⁹ se ha especificado
 - el cursor está definido estáticamente
 - la opción de enlace LANGLEVEL es MIA o SQL92E

Una columna de la lista de selección de la selección completa externa asociada con un cursor es *actualizable* si se cumplen todas las condiciones siguientes:

- el cursor es suprimible
- la resolución de la columna devuelve una columna de la tabla base
- la opción de enlace LANGLEVEL es MIA o SQL92E, o la sentencia-select incluye la cláusula FOR UPDATE (la columna debe estar especificada explícita o implícitamente en la cláusula FOR UPDATE).

Un cursor es de *sólo-lectura* si no es suprimible.

Un cursor es *ambiguo* si se cumplen todas las condiciones siguientes:

- la sentencia-select se prepara de forma dinámica
- la sentencia-select no incluye la cláusula FOR READ ONLY ni la cláusula FOR UPDATE
- la opción de enlace LANGLEVEL es SAA1
- por lo demás, el cursor cumple los requisitos de un cursor suprimible.

Un cursor ambiguo se considera de sólo lectura si la opción de enlace BLOCKING es ALL, de lo contrario se considera actualizable.

- Los cursores de procedimientos almacenados que son invocados por programas de aplicación que se han escrito utilizando CLI pueden utilizarse para definir conjuntos resultantes que se devuelven directamente a la aplicación cliente. Los cursores de procedimientos SQL también pueden devolver resultados a un procedimiento SQL llamador sólo si están definidos utilizando la cláusula WITH RETURN. Vea “Notas” en la página 622.

88. La cláusula FOR READ ONLY está definida en “cláusula-read-only” en la página 527.

89. La cláusula FOR UPDATE se define en el apartado “cláusula-update” en la página 526.

Ejemplo

La sentencia **DECLARE CURSOR** asocia el nombre del cursor C1 con los resultados de **SELECT**.

```
EXEC SQL DECLARE C1 CURSOR FOR  
      SELECT DEPTNO, DEPTNAME, MGRNO  
      FROM DEPARTMENT  
      WHERE ADMRDEPT = 'A00';
```

DECLARE GLOBAL TEMPORARY TABLE

DECLARE GLOBAL TEMPORARY TABLE

La sentencia DECLARE GLOBAL TEMPORARY TABLE define una tabla temporal para la sesión actual. La descripción de la tabla temporal declarada no aparece en el catálogo del sistema. No es una tabla permanente y no se puede compartir con otras sesiones. Cada sesión que define una tabla temporal global declarada del mismo nombre tiene su propia descripción exclusiva de la tabla temporal. Cuando la sesión finaliza, se suprimen las filas de la tabla y se elimina la descripción de la tabla temporal.

Invocación

Esta sentencia se puede intercalar en un programa de aplicación o se puede emitir utilizando sentencias de SQL dinámicas. Es una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización

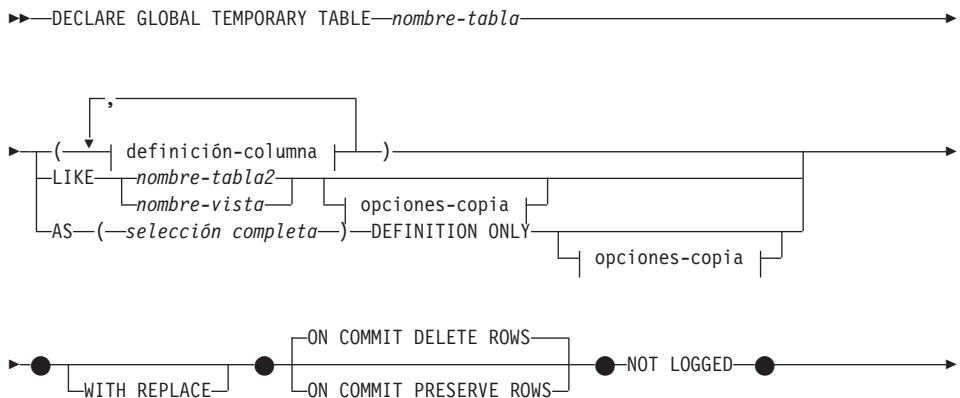
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Privilegio USE para el espacio de tablas USER TEMPORARY.

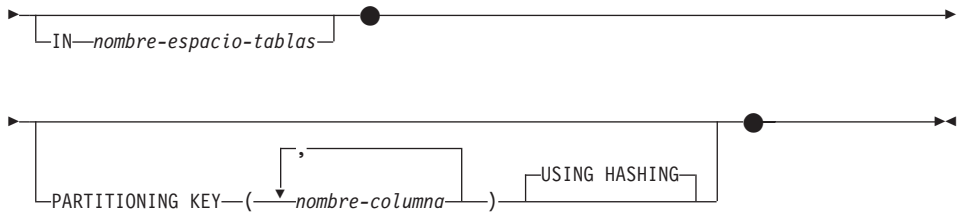
Cuando se define una tabla utilizando LIKE o una selección completa, el ID de autorización de la sentencia debe también tener como mínimo uno de los privilegios siguientes para cada tabla o vista identificada:

- Privilegio SELECT para la tabla o vista
- Privilegio CONTROL para la tabla o vista
- Autorización SYSADM o DBADM

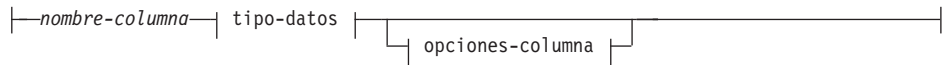
Sintaxis



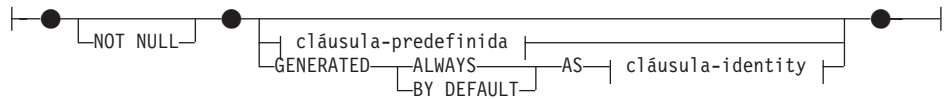
DECLARE GLOBAL TEMPORARY TABLE



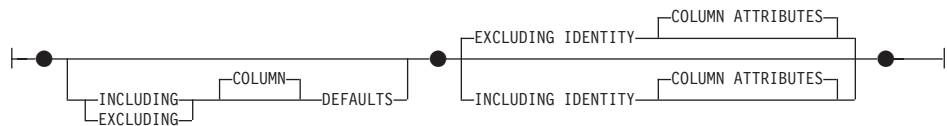
definición-columna:



opciones-columna:



opciones-copia:



Descripción

nombre-tabla

Designa la tabla temporal. Si se especifica explícitamente, el calificador debe ser `SESSION`, de lo contrario se produce un error (SQLSTATE 428EK). Si no se especifica el calificador, se asigna `SESSION` de forma implícita.

Cada sesión que define una tabla temporal global declarada con el mismo *nombre-tabla* tiene su propia descripción exclusiva de esa tabla temporal. La cláusula `WITH REPLACE` se debe especificar si *nombre-tabla* identifica una tabla temporal declarada que ya existe en la sesión (SQLSTATE 42710).

Es posible que el catálogo ya contenga una tabla, vista, seudónimo o apodo que tenga el mismo nombre y el nombre de esquema `SESSION`. En este caso:

DECLARE GLOBAL TEMPORARY TABLE

- Es posible todavía definir una tabla temporal global declarada *nombre-tabla* sin provocar un error ni un aviso
- Las referencias a *SESSION.nombre-tabla* se resolverán en una tabla temporal global declarada, en lugar de en la tabla *SESSION.nombre-tabla* ya definida en el catálogo.

definición-columna

Define los atributos de una columna de la tabla temporal.

nombre-columna

Es el nombre de una columna de la tabla. El nombre no puede estar calificado y no se puede utilizar el mismo nombre para más de una columna de la tabla (SQLSTATE 42711).

Una tabla puede tener las características siguientes:

- un tamaño de página igual a 4K, con un máximo de 500 columnas, donde el total de bytes de las columnas no debe ser mayor que 4005 en un tamaño de página de 4K. Vea “Tamaño de fila” en la página 885 para obtener más detalles.
- un tamaño de página igual a 8K, con un máximo de 1012 columnas, donde el total de bytes de las columnas no debe ser mayor que 8101. Vea “Tamaño de fila” en la página 885 para obtener más detalles.
- un tamaño de página igual a 16K, con un máximo de 1012 columnas, donde el total de bytes de las columnas no debe ser mayor que 16293. Vea “Tamaño de fila” en la página 885 para obtener más detalles.
- un tamaño de página igual a 32K, con un máximo de 1012 columnas, donde el total de bytes de las columnas no debe ser mayor que 32677. Vea “Tamaño de fila” en la página 885 para obtener más detalles.

tipo-datos

Vea *tipo-datos* en “CREATE TABLE” en la página 835 para conocer los tipos permitidos. Los tipos BLOB, CLOB, DBCLOB, LONG VARCHAR, LONG VARGRAPHIC, DATALINK, los tipos de referencia y los tipos estructurados no se pueden utilizar en tablas temporales globales declaradas (SQLSTATE 42962). Esta restricción es aplicable también a los tipos diferenciados basados en esos tipos no permitidos.

Se puede especificar FOR BIT DATA como parte de los tipos de datos de serie de caracteres.

opciones-columna

Define otras opciones relacionadas con las columnas de la tabla.

NOT NULL

Evita que la columna contenga valores nulos. Vea *NOT NULL* en “CREATE TABLE” en la página 835 para conocer detalles sobre la especificación de valores nulos.

cláusula-predefinida

Vea *cláusula-predefinida* en “CREATE TABLE” en la página 835 para conocer detalles sobre la especificación de valores por omisión.

cláusula-identity

Vea *cláusula-identity* en “CREATE TABLE” en la página 835 para conocer detalles sobre la especificación de columnas de identidad.

LIKE *nombre-tabla2* o *nombre-vista*

Especifica que las columnas de la tabla tienen exactamente el mismo nombre y descripción que las columnas de la tabla identificada (*nombre-tabla2*), vista (*nombre-vista*) o apodo (*apodo*). El nombre especificado después de LIKE debe identificar una tabla, vista o apodo existentes en el catálogo, o una tabla temporal declarada. No se puede especificar una tabla con tipo ni una vista con tipo (SQLSTATE 428EC).

El uso de LIKE es una definición implícita de n columnas, donde n es el número de columnas de la tabla o vista identificada.

- Si se designa una tabla, la definición implícita incluye el nombre de columna, el tipo de datos y si es posible la existencia de nulos para cada columna de *nombre-tabla2*. Si no se especifica EXCLUDING COLUMN DEFAULTS, también se incluye el valor por omisión de la columna.
- Si se designa una vista, la definición implícita incluye el nombre de columna, tipo de datos y posibilidad de contener nulos de cada columna resultante de la selección completa definida en *nombre-vista*.

Según cuáles sean las cláusulas de atributos de copia, se pueden incluir o excluir el valor por omisión de la columna y los atributos IDENTITY de la columna.

La definición implícita no incluye otros atributos de la tabla o vista identificada. Por lo tanto, la nueva tabla no tiene ninguna restricción de unicidad, restricción de clave foránea, desencadenantes ni índices. La tabla se crea en el espacio de tablas, de forma implícita o explícita, de acuerdo con lo especificado por la cláusula IN.

Los nombres utilizados para *nombre-tabla2* y *nombre-vista* no pueden ser iguales que el nombre de la tabla temporal global que se está creando (SQLSTATE 428EC).

AS (*selección completa*) DEFINITION ONLY

Especifica si la definición de tabla se basa en las definiciones de columnas

DECLARE GLOBAL TEMPORARY TABLE

procedentes del resultado de una consulta. El uso de AS (*selección completa*) es una definición implícita de n columnas de la tabla temporal global declarada, donde n es el número de columnas resultantes de la *selección completa*. Las columnas de la nueva tabla se definen basándose en las columnas que resultan de la *selección completa*. Cada elemento de la lista de selección debe tener un nombre exclusivo (SQLSTATE 42711). La cláusula AS se puede utilizar la cláusula-select para proporcionar nombres exclusivos.

La definición implícita incluye el nombre de columna, el tipo de datos y la posibilidad de contener nulos de cada columna resultante de la *selección completa*.

opciones-copia

Estas opciones especifican si deben copiarse atributos adicionales de la definición de la tabla resultante original (tabla, vista o selección completa).

INCLUDING COLUMN DEFAULTS

Especifica que deben copiarse los valores por omisión de cada columna actualizable contenida en la definición de la tabla resultante origen. Las columnas que no son actualizables no tienen un valor por omisión definido en la correspondiente columna de la tabla creada.

Si se especifica LIKE *nombre-tabla2* y *nombre-tabla2* designa una tabla base o una tabla temporal declarada, la opción por omisión es INCLUDING COLUMN DEFAULTS.

EXCLUDING COLUMN DEFAULTS

Especifica que no deben copiarse los valores por omisión de columnas contenidos en la definición de la tabla resultante origen.

Esta es la cláusula por omisión, excepto si se especifica LIKE *nombre-tabla* y *nombre-tabla* designa una tabla base o una tabla temporal declarada.

INCLUDING IDENTITY COLUMN ATTRIBUTES

Especifica que deben copiarse, si existen, los atributos de columna de identidad (valores START WITH, INCREMENT BY y CACHE) contenidos en la definición de la tabla resultante origen. Se pueden copiar estos atributos si el elemento de la correspondiente columna de la tabla, vista o selección completa es el nombre de una columna de tabla o de vista que, directa o indirectamente, se correlaciona con el nombre de una columna de tabla base que tiene el atributo de identidad. En todos los demás casos, las columnas de la nueva tabla temporal no tendrán el atributo de identidad. Por ejemplo:

- La lista de selección de la selección completa contiene varias instancias del nombre de una columna de identidad (es decir, se selecciona la misma columna más de una vez)

DECLARE GLOBAL TEMPORARY TABLE

- la lista de selección de la selección completa contiene varias columnas de identidad (es decir, supone la ejecución de una operación de unión)
- la columna de identidad está contenida en una expresión de la lista de selección
- la selección completa contiene una operación de conjuntos (union, except o intersect).

EXCLUDING IDENTITY COLUMN ATTRIBUTES

Especifica que no deben copiarse los atributos de columna de identidad contenidos en la definición de la tabla resultante origen.

ON COMMIT

Especifica la acción que se realiza sobre la tabla temporal global cuando se ejecuta una operación COMMIT.

DELETE ROWS

Se suprimen todas las filas de la tabla si no hay ningún cursor abierto en la tabla que esté definido con WITH HOLD. Éste es el valor por omisión.

PRESERVE ROWS

Las filas de la tabla se conservan.

NOT LOGGED

Los cambios hechos en la tabla no se registran en un archivo de anotaciones, incluida la creación de la tabla. Cuando se realiza una operación ROLLBACK (o ROLLBACK TO SAVEPOINT) y la tabla se ha cambiado en la unidad de trabajo (o punto de salvar), entonces se suprimen todas las filas de la tabla. Si la tabla se creó en la unidad de trabajo (o punto de salvar), entonces se elimina la tabla. Si la tabla se eliminó en la unidad de trabajo (o punto de salvar), entonces se restaura la tabla, pero sin ninguna fila. Además, si una sentencia INSERT, UPDATE o DELETE detecta un error al ejecutarse para la tabla, se suprimen todas las filas de la tabla.

WITH REPLACE

Indica que, si ya existe una tabla temporal global declarada con el nombre especificado, la tabla existente es sustituida por la tabla temporal definida en esta sentencia (y se suprimen todas las filas de la tabla existente).

Si no se especifica WITH REPLACE, el nombre especificado no debe identificar una tabla temporal global declarada que ya exista en la sesión actual (SQLSTATE 42710).

IN *nombre-espacio-tablas*

Identifica el espacio de tablas donde se crearán instancias de la tabla temporal global. El espacio de tablas debe existir y estar definido como USER TEMPORARY (SQLSTATE 42838), para el cual el ID de autorización

DECLARE GLOBAL TEMPORARY TABLE

de la sentencia tiene el privilegio USE (SQLSTATE 42501). Si no se especifica esta cláusula, se elige el espacio de tablas USER TEMPORARY con el tamaño de página menor posible para el cual el ID de autorización de la sentencia tenga el privilegio USE. Cuando existe más de un espacio de tablas que puede elegirse, se establece una prioridad basada en quién recibió el privilegio USE:

1. el ID de autorización
2. un grupo al cual pertenece el ID de autorización
3. PUBLIC

Si todavía existe más de un espacio de tablas que puede elegirse, el gestor de bases de datos toma la decisión final. Cuando no hay ninguna tabla USER TEMPORARY elegible, se produce un error (SQLSTATE 42727).

La determinación del espacio de tablas puede cambiar cuando:

- se eliminan o crean espacios de tablas
- se otorgan o revocan privilegios USE.

El tamaño de página suficiente de una tabla está determinado por el número de bytes de la fila o por el número de columnas. Consulte el apartado “Tamaño de fila” en la página 885 para obtener más información.

PARTITIONING KEY (*nombre-columna,...*)

Especifica la clave de particionamiento utilizada cuando se particionan los datos de la tabla. Cada *nombre-columna* debe identificar una columna de la tabla y la misma columna no debe estar identificada más de una vez.

Si no se especifica esta cláusula y la tabla reside en un grupo de nodos de varias particiones, la clave de particionamiento se define como la primera columna de la tabla temporal declarada.

Para las tablas temporales declaradas, que residen en espacios de tablas definidos en grupos de nodos de una sola partición, puede utilizarse cualquier conjunto de columnas para definir la clave de particionamiento. Si no especifica este parámetro, no se crea ninguna clave de particionamiento.

Las columnas de clave de particionamiento pueden actualizarse, a no ser que el parámetro de configuración DB2_UPDATE_PART_KEY se haya establecido en 'OFF' (SQLSTATE 42997).

USING HASHING

Especifica la utilización de la función de generación aleatoria como el método de partición para la distribución de datos. Es el único método de partición soportado.

Notas

- **Referencia a una tabla temporal global declarada:** La descripción de una tabla temporal global declarada no aparece en el catálogo de DB2 (SYSCAT.TABLES); por consiguiente, no es permanente y no se puede compartir con otras conexiones a la base de datos. Esto significa que cada sesión que define una tabla temporal global declarada (*nombre-tabla*) tiene su propia descripción exclusiva de esa tabla.

Para hacer referencia a la tabla temporal global declarada en una sentencia de SQL (distinta de la sentencia DECLARE GLOBAL TEMPORARY TABLE), la tabla debe estar calificada, implícita o explícitamente, por el nombre de esquema SESSION. Si *nombre-tabla* no está calificado por SESSION, las tablas temporales globales declaradas no se tienen en cuenta al resolver la referencia.

Una referencia a SESSION.*nombre-tabla* en una conexión que no ha declarado una tabla temporal global mediante ese nombre intentará realizar la resolución a partir de objetos permanentes del catálogo. Si dicho objeto no existe, se produce un error (SQLSTATE 42704).

- Cuando se enlaza un paquete que tiene sentencias de SQL estáticas que hacen referencia a tablas calificadas, implícita o explícitamente, por SESSION, esas sentencias no se enlazarán de forma estática. Cuando se invocan estas sentencias, se enlazan de forma incremental, cualquiera que sea la opción de VALIDATE elegida al enlazar el paquete. Durante la ejecución, la resolución de cada referencia a tabla da como resultado una tabla temporal declarada o una tabla permanente. Si no existe ninguna de las dos, se produce un error (SQLSTATE 42704).
- **Privilegios:** Cuando se define una tabla temporal global declarada, el definidor de la tabla recibe todos los privilegios para la tabla, incluida la capacidad para eliminar la tabla. Además, estos privilegios se otorgan a PUBLIC.⁹⁰ Esto permite que cualquier sentencia de SQL de la sesión haga referencia a una tabla temporal global declarada que ya se ha definido en esa sesión.
- **Creación de instancias y terminación:** En lo que respecta a la explicación que sigue a continuación, P representa una sesión y T es una tabla temporal global declarada de la sesión P:
 - La sentencia DECLARE GLOBAL TEMPORARY TABLE que se ejecuta en P crea una instancia vacía de T.
 - Cualquier sentencia de SQL de P puede hacer referencia a T, y cualquier referencia a T en P es una referencia a esa misma instancia de T.
 - Si se especifica una sentencia DECLARE GLOBAL TEMPORARY TABLE dentro de la sentencia compuesta del procedimiento SQL (definida por BEGIN y END), el ámbito de la tabla temporal global declarada es la

90. Ninguno de los privilegios se otorga con la opción GRANT y ninguno aparece en la tabla de catálogo.

DECLARE GLOBAL TEMPORARY TABLE

conexión, no sólo la sentencia compuesta, y la tabla es conocida fuera de la sentencia compuesta. La tabla no se elimina implícitamente al final de la sentencia compuesta. Una tabla temporal global declarada no se puede definir varias veces con el mismo nombre en otras sentencias compuestas de la sesión, a menos que se haya eliminado explícitamente la tabla.

- Si se ha especificado, implícita o explícitamente, la cláusula ON COMMIT DELETE ROWS, cuando una operación de confirmación finaliza una unidad de trabajo en P, y no existe ningún cursor abierto en P que esté definido con WITH HOLD y dependa de T, la confirmación incluye la operación DELETE FROM SESSION.T.

- Cuando una operación de retrotracción finaliza una unidad de trabajo o un punto de salvar en P, y esa unidad de trabajo o punto de salvar incluye una modificación en SESSION.T, la retrotracción incluye la operación DELETE from SESSION.T.

Cuando una operación de retrotracción finaliza una unidad de trabajo o un punto de salvar en P, y esa unidad de trabajo o punto de salvar incluye la declaración de SESSION.T, la retrotracción incluye la operación DROP SESSION.T.

Cuando una operación de retrotracción finaliza una unidad de trabajo o un punto de salvar en P, y esa unidad de trabajo o punto de salvar incluye la eliminación de la tabla temporal declarada SESSION.T, la retrotracción deshace la eliminación de la tabla, pero la tabla estará vacía.

- Cuando el proceso de aplicación donde se declaró T finaliza o se desconecta de la base de datos, se elimina T y se destruyen las instancias de sus filas.
- Cuando finaliza la conexión con el servidor donde se declaró T, se elimina T y se destruyen las instancias de sus filas.

- **Restricciones sobre la utilización de tablas temporales globales declaradas:** Las tablas temporales globales declaradas:

- No se pueden especificar en las sentencias ALTER, COMMENT, GRANT, LOCK, RENAME ni REVOKE (SQLSTATE 42995).
- No pueden estar referenciadas en las sentencias CREATE ALIAS, CREATE FUNCTION (escalar SQL, de tabla o de fila), CREATE INDEX, CREATE TRIGGER ni CREATE VIEW (SQLSTATE 42995).
- No se pueden especificar en restricciones de referencia (SQLSTATE 42995).

DELETE

La sentencia DELETE suprime filas de una tabla o vista. La supresión de una fila de una vista suprime la fila de la tabla en la que se basa la vista.

Existen dos formas de esta sentencia:

- La forma DELETE *con búsqueda* se utiliza para suprimir una o varias filas (determinadas opcionalmente mediante una condición de búsqueda).
- La forma DELETE *con posición* se utiliza para suprimir una fila exactamente (determinada por la posición actual del cursor).

Invocación

Una sentencia DELETE puede estar incorporada en un programa de aplicación o puede emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización

Para ejecutar cualquiera de las dos formas de esta sentencia, el ID de autorización de la sentencia debe poseer como mínimo los siguientes privilegios:

- El privilegio DELETE en la tabla o vista de las que se han de suprimir las filas
- El privilegio CONTROL en la tabla o vista de las que se han de suprimir las filas
- Autorización SYSADM o DBADM.

Para ejecutar la sentencia DELETE con búsqueda, los privilegios del ID de autorización de la sentencia también deben incluir como mínimo uno de los siguientes para cada tabla o vista a la que se hace referencia en una subconsulta:

- Privilegio SELECT
- Privilegio CONTROL
- Autorización SYSADM o DBADM.

Cuando se precompila el paquete con reglas SQL92 ⁹¹ y la forma de una sentencia DELETE con búsqueda incluye una referencia a una columna de la tabla o vista en la *condición-búsqueda*, los privilegios retenidos por el ID de autorización de la sentencia también deben incluir como mínimo uno de los siguientes:

- Privilegio SELECT

91. El paquete que se utiliza para procesar la sentencia se precompila utilizando la opción LANGLEVEL con el valor SQL92E o MIA.

DELETE

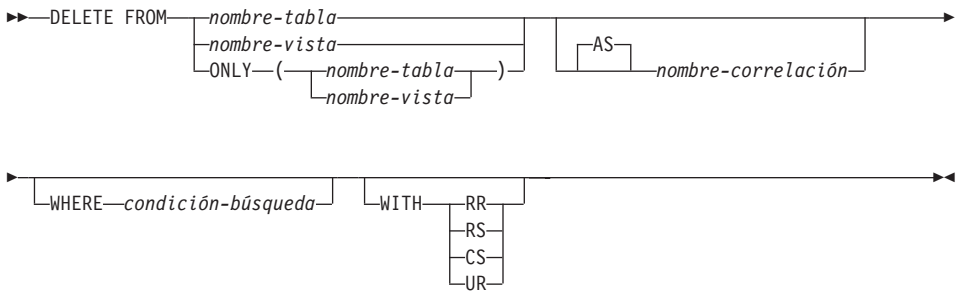
- Privilegio CONTROL
- Autorización SYSADM o DBADM.

Cuando la vista o tabla especificada está precedida de la palabra clave ONLY, los privilegios que tiene el ID de autorización de la sentencia también deben incluir el privilegio SELECT para cada subvista o subtabla de la vista o tabla especificada.

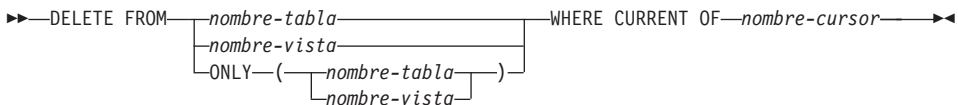
Los privilegios de grupo no se comprueban para las sentencias DELETE estáticas.

Sintaxis

DELETE de búsqueda:



DELETE con posición:



Descripción

FROM nombre-tabla o nombre-vista

Identifica la tabla o vista en las que se han de suprimir las filas. El nombre debe identificar una tabla o vista que exista en el catálogo, pero no debe identificar una tabla del catálogo, una vista de catálogo, una tabla de resumen ni una vista de sólo lectura. (Para obtener una explicación de las vistas de sólo lectura, consulte el apartado "CREATE VIEW" en la página 957.)

Si *nombre-tabla* es una tabla con tipo, la sentencia puede suprimir filas de la tabla o cualquiera de sus subtablas correspondientes.

Si *nombre-vista* es una vista con tipo, la sentencia puede que elimine las filas de la vista subyacente o de las vistas subyacentes de las subvistas correspondientes de la vista. Si *nombre-vista* es una vista regular con una tabla subyacente que es una tabla con tipo, la sentencia puede suprimir filas de la tabla con tipo o cualquiera de sus propias subtablas.

Sólo puede hacerse referencia a las columnas de la tabla especificada en la cláusula WHERE. Para una sentencia DELETE con posición, el cursor asociado también debe haber especificado la tabla o vista en la cláusula FROM sin utilizar ONLY.

FROM ONLY (*nombre-tabla*)

Aplicable a las tablas con tipo, la palabra clave ONLY especifica que la sentencia sólo se debe aplicar a los datos de la tabla especificada y no puede suprimir las filas de las subtablas correspondientes. Para una sentencia DELETE con posición, el cursor asociado también debe haber especificado la tabla en la cláusula FROM utilizando ONLY. Si *nombre-tabla* no es una tabla con tipo, la palabra clave ONLY no tiene ningún efecto en la sentencia.

FROM ONLY (*nombre-vista*)

Aplicable a las vistas con tipo, la palabra clave ONLY especifica que la sentencia sólo se debe aplicar a los datos de la vista especificada y no puede suprimir las filas de las subvistas correspondientes. Para una sentencia DELETE con posición, el cursor asociado también debe haber especificado la vista en la cláusula FROM utilizando ONLY. Si *nombre-vista* no es una vista con tipo, la palabra clave ONLY no tiene ningún efecto en la sentencia.

nombre-correlación

Se puede utilizar dentro de la condición-búsqueda para designar la tabla o vista. (Para obtener una explicación del nombre-correlación, consulte el “Capítulo 3. Elementos del lenguaje” en la página 69.)

WHERE

Especifica una condición que selecciona las filas que se han de suprimir. Puede omitirse la cláusula, se puede especificar una condición de búsqueda o nombrar un cursor. Si se omite la cláusula, se suprimen todas las filas de la tabla o vista.

condición-búsqueda

Es cualquier condición de búsqueda descrita en el apartado “Condiciones de búsqueda” en la página 232. Cada *nombre-columna* de la condición de búsqueda, que no sea na subconsulta, debe identificar una columna de la tabla o vista.

La *condición-búsqueda* se aplica a cada fila de la tabla o vista y las filas suprimidas son aquellas para las que el resultado de la *condición-búsqueda* es verdadero.

DELETE

Si la condición de búsqueda contiene una subconsulta, puede considerarse que ésta se ejecuta cada vez que se aplica la *condición de búsqueda* a una fila y que los resultados se utilizan para aplicar la *condición de búsqueda*. De hecho, una subconsulta sin referencias correlacionadas sólo se ejecuta una vez, mientras que una subconsulta con una referencia correlacionada puede tener que ejecutarse una vez para cada fila. Si una subconsulta hace referencia a una tabla de objetos de una sentencia DELETE o a una tabla dependiente con una regla de supresión de CASCADE o SET NULL, la subconsulta se evalúa por completo antes de suprimir cualquier fila.

CURRENT OF *nombre-cursor*

Identifica un cursor que se define en una sentencia DECLARE CURSOR del programa. La sentencia DECLARE CURSOR debe preceder a la sentencia DELETE.

La tabla o vista nombradas también deben nombrarse en la cláusula FROM de la sentencia SELECT del cursor, y la tabla resultante del cursor no debe ser de sólo lectura. (Para ver una explicación de las tablas resultantes de sólo lectura, consulte el apartado “DECLARE CURSOR” en la página 976.)

Cuando se ejecuta la sentencia DELETE, el cursor debe posicionarse en una fila: dicha fila es la que se suprime. Después de la supresión, el cursor se posiciona antes de la siguiente fila de la tabla resultante. Si no hay una fila siguiente, el cursor se posiciona después de la última fila.

WITH

Especifica el nivel de aislamiento utilizado al localizar las filas que se deben suprimir.

RR

Lectura repetible

RS

Estabilidad de lectura

CS

Estabilidad del cursor

UR

Lectura no confirmada

El nivel de aislamiento por omisión de la sentencia es el nivel de aislamiento del paquete en el que está enlazada la sentencia.

Normas

- Si la tabla identificada o la tabla base de la vista identificada es padre, las filas seleccionadas para supresión no deben tener ningún dependiente en

una relación con una regla de supresión de RESTRICT, y DELETE no debe aplicarse en cascada a las filas descendientes que tengan dependientes en una relación con una regla de supresión de RESTRICT.

Si no se impide la operación de supresión por una regla de supresión RESTRICT, se suprimen las filas seleccionadas. Las fila que son dependientes de la filas seleccionadas también se ven afectadas:

- Las columnas con posibilidad de contener nulos de las claves foráneas de cualquier fila que sea dependiente en una relación con una regla de supresión de SET NULL se establecen en el valor nulo.
- Las filas que sean sus dependientes en una relación con una regla de supresión de CASCADE también se suprimen y se aplican, a su vez, las reglas anteriores a estas filas.

Se comprueba la regla de supresión de NO ACTION para imponer que las claves foráneas no nulas hagan referencia a una fila padre existente después de que se hayan impuesto otras restricciones de referencia.

Notas

- Si se produce un error durante la ejecución de una sentencia DELETE de múltiples filas, no se realiza ningún cambio en la base de datos.
- A menos de que ya existan los bloqueos adecuados, se adquieren uno o más bloqueos exclusivos durante la ejecución de una sentencia DELETE satisfactoria. La emisión de una sentencia COMMIT o ROLLBACK liberará los bloqueos. Hasta que se liberen los bloqueos por una operación de confirmación o de retroacción, el efecto de la operación de supresión sólo lo percibirán:
 - El proceso de aplicación que ha realizado la supresión
 - Otro proceso de aplicación que utilice el nivel de aislamiento UR.

Los bloqueos pueden evitar que otros procesos de aplicación realicen operaciones en la tabla.

- Si un proceso de aplicación suprime una fila en la que está posicionado alguno de sus cursores, dichos cursores se posicionan antes de la siguiente fila de la tabla resultante. Supongamos que C sea un cursor que está situado antes de la fila R (como resultado de una operación OPEN, una operación DELETE a través de C, una operación DELETE a través de otro cursor o una operación DELETE con búsqueda). Si existen operaciones INSERT, UPDATE y DELETE que afectan a la tabla base de la que deriva R, la siguiente operación FETCH que haga referencia a C no posiciona necesariamente C en R. Por ejemplo, la operación puede posicionar C en R', donde R' es una nueva fila que ahora es la fila siguiente de la tabla resultante.
- SQLERRD(3) en la SQLCA muestra el número de filas suprimidas de la tabla de objetos después de la ejecución de la sentencia. No incluye las filas que se han suprimido como resultado de una regla de supresión

DELETE

CASCADE. SQLERRD(5) en la SQLCA muestra el número de filas afectadas por las restricciones de referencia y por las sentencias activadas. Incluye filas que se han suprimido como resultado de una regla de supresión CASCADE y filas en las que se han establecido claves foráneas en NULL como resultado de una regla de supresión SET NULL. En relación a la sentencias activadas, incluye el número de filas que se han insertado, actualizado o suprimido. (Para ver una descripción de la SQLCA, consulte el “Apéndice B. Comunicaciones SQL (SQLCA)” en la página 1261.)

- Si se produce un error que impide la supresión de todas las filas que coincidan con la condición de búsqueda y todas las operaciones necesarias para las restricciones de referencia existentes, no se realiza ningún cambio en la tabla y se devuelve un error.
- En cualquier fila suprimida que incluyera archivos actualmente enlazados mediante columnas DATALINK, los archivos se desenlazan y se restaurarán o se suprimirán según la definición de la columna DATALINK.

Puede producirse un error cuando se intente suprimir un valor DATALINK si el servidor de archivos del valor ya no está registrado con el servidor de bases de datos (SQLSTATE 55022).

También puede producirse un error cuando se suprima una fila que tenga un enlace con un servidor que no esté disponible en el momento de la supresión (SQLSTATE 57050).

Ejemplos

Ejemplo 1: Suprima el departamento (DEPTNO) 'D11' de la tabla DEPARTMENT.

```
DELETE FROM DEPARTMENT
WHERE DEPTNO = 'D11'
```

Ejemplo 2: Suprima todos los departamentos de la tabla DEPARTMENT (es decir, vacíe la tabla).

```
DELETE FROM DEPARTMENT
```

Ejemplo 3: Suprima de la tabla EMPLOYEE los representantes de ventas o los representantes de zona que no hayan realizado ninguna venta en 1995.

```
DELETE FROM EMPLOYEE
WHERE LASTNAME NOT IN
(SELECT SALES_PERSON
FROM SALES WHERE YEAR(SALES_DATE)=1995)
AND JOB IN ('SALESREP','FIELDREP')
```

DESCRIBE

La sentencia DESCRIBE obtiene información acerca de una sentencia preparada. Para obtener una explicación de las sentencias preparadas, consulte el apartado "PREPARE" en la página 1099.

Invocación

Esta sentencia sólo se puede incluir en un programa de aplicación. Se trata de una sentencia ejecutable que no puede prepararse de forma dinámica.

Autorización

No se necesita.

Sintaxis

►►—DESCRIBE—*nombre-sentencia*—INTO—*nombre-descriptor*—►►

Descripción

nombre-sentencia

Identifica la sentencia sobre la que se necesita información. Cuando se ejecuta la sentencia DESCRIBE, el nombre debe identificar una sentencia preparada.

INTO *nombre-descriptor*

Identifica un área de descriptores SQL (SQLDA), que se describen en el "Apéndice C. Área de descriptores SQL (SQLDA)" en la página 1267. Antes de ejecutar la sentencia DESCRIBE, deben establecerse las siguientes variables en la SQLDA:

SQLN Indica el número de variables representadas por SQLVAR. (SQLN proporciona la dimensión de la matriz SQLVAR.) SQLN debe establecerse en un valor mayor o igual que cero antes de ejecutar la sentencia DESCRIBE.

Cuando se ejecuta la sentencia DESCRIBE, el gestor de bases de datos asigna valores a las variables de la SQLDA de la siguiente manera:

SQLDAID

Los 6 primeros bytes se establecen en 'SQLDA ' (es decir, 5 letras seguidas del carácter de espacio).

El séptimo byte, llamado SQLDOUBLED, se establece en '2' si la SQLDA contiene dos entradas SQLVAR para cada elemento de la lista-selección (o la *columna* de la tabla resultante). Esta técnica se utiliza para acomodar columnas resultantes de tipo LOB, de tipo diferenciado, de tipo estructurado o de tipo de referencia. De lo contrario, SQLDOUBLED se establece en el carácter de espacio.

DESCRIBE

Se establece el doble distintivo en espacio si no hay suficiente espacio en la SQLDA para contener toda la respuesta DESCRIBE.

El octavo byte se establece en el carácter de espacio.

SQLDABC

Longitud de la SQLDA.

SQLD Si la sentencia preparada es SELECT, el número de columnas de su tabla resultante; de lo contrario, 0.

SQLVAR

Si el valor de SQLD es 0 o mayor que el valor de SQLN, no se asigna ningún valor a las ocurrencias de SQLVAR.

Si el valor es n , donde n es mayor que 0 pero menor o igual al valor de SQLN, los valores se asignan a las n primeras ocurrencias de SQLVAR para que la primera ocurrencia de SQLVAR contenga una descripción de la primera columna de la tabla resultante, la segunda ocurrencia de SQLVAR contenga una descripción de la segunda columna de la tabla resultante, etcétera. La descripción de una columna consta de los valores asignados a SQLTYPE, SQLLEN, SQLNAME, SQLLONGLEN y SQLDATATYPE_NAME.

SQLVAR básica

SQLTYPE

Un código que muestra el tipo de datos de la columna y si puede contener valores nulos o no.

SQLLEN

Un valor de longitud que depende del tipo de datos de las columnas del resultado. SQLLEN es 0 para tipos de datos LOB.

SQLNAME

Si la columna derivada no es una referencia de columna simple, sqlname contiene un valor de literal numérico ASCII, que representa la posición original de la columna derivada dentro de la lista de selección; de lo contrario, sqlname contiene el nombre de la columna.

SQLVAR secundaria

Estas variables sólo se utilizan si el número de entradas SQLVAR se doblan para acomodar columnas de tipo LOB, de tipo diferenciado, de tipo estructurado o de tipo de referencia.

SQLLONGLEN

El atributo de longitud de una columna BLOB, CLOB o DBCLOB.

SQLDATATYPE_NAME

Para cualquier columna de tipo definido por el usuario (diferenciado o estructurado), el gestor de bases de datos establece esta opción en el nombre del tipo definido por el usuario, calificado al completo. Para una columna de tipo de referencia, el gestor de bases de datos establece esta opción en el nombre definido por el usuario, calificado al completo, del tipo destino de la referencia. De lo contrario, el nombre de esquema es SYSIBM y el nombre de tipo es el nombre que aparece en la columna TYPENAME de la vista de catálogo SYSCAT.DATATYPES.

Notas

- Antes de ejecutar la sentencia DESCRIBE, el valor de SQLN debe establecerse de manera que indique cuántas ocurrencias de SQLVAR se proporcionan en la SQLDA y debe asignarse suficiente almacenamiento para contener ocurrencias de SQLN. Para obtener la descripción de las columnas de la tabla resultante de una sentencia SELECT preparada, el número de ocurrencias de SQLVAR no debe ser menor que el número de columnas.
- Si se espera un LOB de gran tamaño, tenga en cuenta que el manejo de este LOB afectará a la memoria de la aplicación. Si se da esta condición, considere la posibilidad de utilizar localizadores o variables de referencia a archivos. Modifique la SQLDA después de ejecutar la sentencia DESCRIBE pero antes de asignar almacenamiento para que un SQLTYPE de SQL_TYP_xLOB se cambie a SQL_TYP_xLOB_LOCATOR o SQL_TYP_xLOB_FILE con los cambios correspondientes a otros campos como, por ejemplo, SQLLEN. Después asigne el almacenamiento basado en SQLTYPE y continúe.

Consulte el manual *Application Development Guide* para obtener más información acerca de la utilización de localizadores y variables de referencia a archivos con la SQLDA.

- Las conversiones de páginas de códigos entre el código Unix ampliado (EUC) y las páginas de códigos DBCS pueden dar como resultado la expansión y contracción de las longitudes de caracteres. Consulte el manual *Application Development Guide* para obtener información acerca del manejo de dichas situaciones.
- Si se selecciona un tipo estructurado, pero no se define ninguna transformación FROM SQL (porque no se especificó ningún TRANSFORM GROUP utilizando el registro especial CURRENT DEFAULT TRANSFORM GROUP (SQLSTATE 428EM), o porque el grupo mencionado no tiene una función de transformación FROM SQL definida (SQLSTATE 42744)), DESCRIBE emitirá un error.

DESCRIBE

- **Asignación de la SQLDA:** Entre las maneras posibles de asignar la SQLDA están las tres descritas abajo.

Primera técnica: Asigne una SQLDA con suficientes ocurrencias de SQLVAR para acomodar cualquier lista de selección que la aplicación vaya a tener que procesar. Si la tabla contiene cualquier columna de tipo LOB, tipo diferenciado, tipo estructurado o de tipo de referencia, el número de las SQLVAR debe ser el doble que el número máximo de columnas; de lo contrario, el número debe ser igual al número máximo de columnas. Después de realizar la asignación, la aplicación puede utilizar esta SQLDA repetidas veces.

Esta técnica utiliza una gran cantidad de almacenamiento que nunca se desasigna, incluso cuando la mayor parte de su almacenamiento no se utilice para una lista de selección en particular.

Segunda técnica: Repita los dos siguientes pasos para cada lista de selección procesada:

1. Ejecute una sentencia DESCRIBE con una SQLDA que no tenga ninguna ocurrencia de SQLVAR; es decir, una SQLDA cuyo SQLN sea cero. El valor devuelto para SQLD es el número de columnas de la tabla resultante. Este es el número necesario de ocurrencias de SQLVAR o la mitad del número necesario. Puesto que no había ninguna entrada SQLVAR, se emitirá un aviso con SQLSTATE 01005. Si el SQLCODE asociado al aviso es +237, +238 ó +239, el número de entradas SQLVAR debe ser el doble que el valor devuelto en SQLD.⁹²
2. Asigne una SQLDA con suficientes ocurrencias de SQLVAR. Después ejecute la sentencia DESCRIBE de nuevo, utilizando esta SQLDA nueva.

Esta técnica permite una mejor gestión del almacenamiento que la primera técnica, pero dobla el número de sentencias DESCRIBE.

Tercera técnica: Asigne una SQLDA que sea lo suficientemente grande para manejar la mayoría de, y quizá todas, las listas de selección pero que también sea razonablemente pequeña. Ejecute DESCRIBE y compruebe el valor SQLD. Utilice el valor SQLD para el número de ocurrencias de SQLVAR para asignar una SQLDA mayor, si es necesario.

Esta técnica está comprendida entre las dos primeras técnicas. Su eficacia depende de una buena elección del tamaño de la SQLDA original.

Ejemplo

En un programa C, ejecute una sentencia DESCRIBE con una SQLDA que no tenga ninguna ocurrencia de SQLVAR. Si SQLD es mayor que cero, utilice el

92. La devolución de estos SQLCODE positivos supone que el valor de la opción de enlace SQLWARN era YES (devuelve SQLCODE positivos). Si SQLWARN estaba establecido en NO, se sigue devolviendo +238 para indicar que el número de entradas SQLVAR debe ser el doble del valor devuelto en SQLD.

valor para asignar una SQLDA con el número necesario de ocurrencias de SQLVAR y después ejecute una sentencia DESCRIBE que utilice dicha SQLDA.

```

EXEC SQL BEGIN DECLARE SECTION;
      char  stmt1_str[200];
EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLDA;
EXEC SQL DECLARE DYN_CURSOR CURSOR FOR STMT1_NAME;

... /* código para solicitar una consulta al usuario, después generar */
      /* una sentencia-select en la stmt1_str          */
EXEC SQL PREPARE STMT1_NAME FROM :stmt1_str;

... /* código para establecer SQLN en cero y asignar la SQLDA */
EXEC SQL DESCRIBE STMT1_NAME INTO :sqlda;

... /* código para comprobar que SQLD se mayor que cero, para establecer */
      /* SQLN en SQLD, después para volver a asignar SQLDA          */
EXEC SQL DESCRIBE STMT1_NAME INTO :sqlda;

... /* código para preparar la utilización de SQLDA          */
      /* y asignar almacenamientos intermedios para recibir datos */
EXEC SQL OPEN DYN_CURSOR;

... /* bucle para leer filas de la tabla resultante
*/
EXEC SQL FETCH DYN_CURSOR USING DESCRIPTOR :sqlda;
.
.
.

```

DISCONNECT

DISCONNECT

La sentencia DISCONNECT finaliza una o varias conexiones cuando no hay ninguna unidad de trabajo activa (es decir, después de una operación de confirmación o retrotracción).⁹³

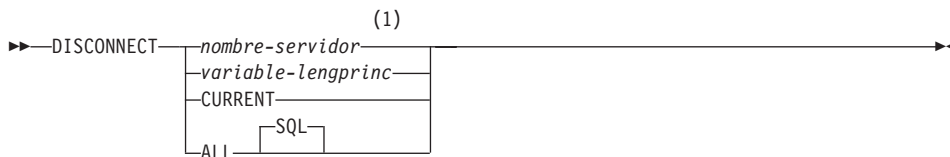
Invocación

Aunque un recurso SQL interactivo pueda proporcionar una interfaz que dé la apariencia de ejecución interactiva, esta sentencia sólo puede incluirse dentro de un programa de aplicación. Se trata de una sentencia ejecutable que no puede prepararse de forma dinámica.

Autorización

No se necesita ninguna.

Sintaxis



Notas:

- 1 Observe que un servidor de aplicaciones llamado CURRENT o ALL sólo puede identificarse mediante una variable del lenguaje principal.

Descripción

nombre-servidor o *variable-lengprinc*

Identifica el servidor de aplicaciones mediante el *nombre-servidor* especificado o mediante la *variable-lengprinc* donde está contenido el *nombre-servidor*.

Si se especifica una *variable-lengprinc*, debe ser una variable de serie de caracteres con un atributo de longitud no superior a 8 y no debe contener una variable indicadora. El *nombre-servidor* contenido en la *variable-lengprinc* debe estar justificado por la izquierda y no estar delimitado por comillas.

Observe que el *nombre-servidor* es un seudónimo de base de datos que identifica al servidor de aplicaciones. Debe estar listado en el directorio local del petionario de aplicaciones.

93. Si la sentencia DISCONNECT se emite para una conexión individual, la conexión sólo finaliza si la base de datos ha participado en cualquier unidad de trabajo existente, con independencia de si hay una unidad de trabajo activa. Por ejemplo, si otras bases de datos han realizado tareas pero el destino en cuestión no lo ha hecho, todavía puede desconectarse sin finalizar la conexión.

El seudónimo-basedatos especificado o el seudónimo-basedatos contenido en la variable del lenguaje principal debe identificar una conexión existente del proceso de aplicación. Si el seudónimo-basedatos no identifica ninguna conexión existente, se genera un error (SQLSTATE 08003).

CURRENT

Identifica la conexión actual del proceso de aplicación. El proceso de aplicación debe estar en el estado conectado. Si no se genera un error (SQLSTATE 08003).

ALL

Indica que se han de destruir todas las conexiones existentes del proceso de aplicación. No se produce ningún error ni aviso si no existen conexiones cuando se ejecuta la sentencia. La palabra clave opcional SQL se incluye para ser coherente con la sintaxis de la sentencia RELEASE.

Normas

- Generalmente, la sentencia DISCONNECT no se puede ejecutar mientras se está en una unidad de trabajo. Si se intenta, se genera un error (SQLSTATE 25000). La excepción a esta regla es si se especifica una sola conexión que se haya de desconectar y la base de datos no ha participado en una unidad de trabajo existente. En este caso, no importa si hay una unidad de trabajo activa cuando se emite la sentencia DISCONNECT.
- La sentencia DISCONNECT no se puede ejecutar nunca en el entorno del Supervisor del Proceso de transacción (TP) (SQLSTATE 25000). Si se utiliza cuando la opción del precompilador es SYNCPOINT se establece en TWOPHASE.

Notas

- Si la sentencia DISCONNECT se realiza satisfactoriamente, se destruye cada conexión.

Si la sentencia DISCONNECT no es satisfactoria, el estado de la conexión del proceso de aplicación y los estados de sus conexiones no se cambian.

- Si se utiliza DISCONNECT para destruir la conexión actual, la siguiente sentencia de SQL ejecutada debe ser CONNECT o SET CONNECTION.
- La semántica de CONNECT Tipo 1 no excluye la utilización de DISCONNECT. Sin embargo, aunque se puedan utilizar DISCONNECT CURRENT y DISCONNECT ALL, no dan como resultado una operación de confirmación como lo haría la sentencia CONNECT RESET.

Si se especifica *nombre-servidor* o *variable-lengprinc* en la sentencia DISCONNECT, debe identificar la conexión actual porque CONNECT Tipo 1 sólo da soporte a una conexión cada vez. Generalmente, DISCONNECT caerá dentro de una unidad de trabajo con la excepción señalada en "Reglas".

DISCONNECT

- Es necesario que los recursos creen y mantengan conexiones remotas. Por lo tanto, una conexión remota que no se vaya a volver a utilizar debe destruirse lo más pronto posible.
- Las conexiones también se pueden destruir durante una operación de confirmación porque la opción de conexión esté en vigor. La opción de conexión podría ser **AUTOMATIC**, **CONDITIONAL** o **EXPLICIT**, que puede establecerse como una opción del precompilador o a través de la API **SET CLIENT** en tiempo de ejecución. Consulte el apartado “Opciones que controlan la semántica de la unidad de trabajo distribuida” en la página 54 para obtener información acerca de la especificación de la opción **DISCONNECT**.

Ejemplos

Ejemplo 1: La aplicación ya no necesita la conexión SQL con **IBMSTHDB**. Debe ejecutarse la sentencia siguiente después de una operación de confirmación o de retrotracción para destruir la conexión.

```
EXEC SQL DISCONNECT IBMSTHDB;
```

Ejemplo 2: La aplicación ya no necesita la conexión actual. Debe ejecutarse la sentencia siguiente después de una operación de confirmación o de retrotracción para destruir la conexión.

```
EXEC SQL DISCONNECT CURRENT;
```

Ejemplo 3: La aplicación ya no necesita las conexiones existentes. Debe ejecutarse la sentencia siguiente después de una operación de confirmación o de retrotracción para destruir todas las conexiones.

```
EXEC SQL DISCONNECT ALL;
```

DROP

La sentencia DROP suprime un objeto. Cualquier objeto que sea dependiente directa o indirectamente de dicho objeto se suprime o pasa a estar no operativo. (Consulte el apartado “Desencadenante no operativo” en la página 917 y el apartado “Vistas no operativas” en la página 967 para obtener detalles.) Siempre que se suprime un objeto, se suprime su descripción del catálogo y se invalidan los paquetes que hacen referencia al objeto.

Invocación

Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

Autorización

El ID de autorización de la sentencia DROP debe incluir uno de los privilegios siguientes cuando se eliminan objetos que admiten nombres de dos partes; de lo contrario se produce un error (SQLSTATE 42501):

- Autorización SYSADM o DBADM
- Privilegio DROPIN para el esquema del objeto
- Definidor del objeto, tal como está registrado en la columna DEFINER de la vista de catálogo del objeto
- Privilegio CONTROL para el objeto (aplicable sólo a índices, especificaciones de índices, apodos, paquetes, tablas y vistas).
- Definidor del tipo definido por el usuario, tal como está registrado en la columna DEFINER de la vista de catálogo SYSCAT.DATATYPES (sólo aplicable al eliminar un método asociado a un tipo definido por el usuario)

El ID de autorización de la sentencia DROP para eliminar una jerarquía de tablas o de vistas debe tener uno de los privilegios anteriores para cada tabla o vista de la jerarquía.

Cuando se elimina un esquema, el ID de autorización de la sentencia DROP debe tener la autorización SYSADM o DBADM, o ser el propietario de esquema tal como está registrado en la columna OWNER de SYSCAT.SCHEMATA.

Cuando se elimina una agrupación de almacenamientos intermedios, un grupo de nodos o un espacio de tablas, el ID de autorización de la sentencia DROP debe tener la autorización SYSADM o SYSCTRL.

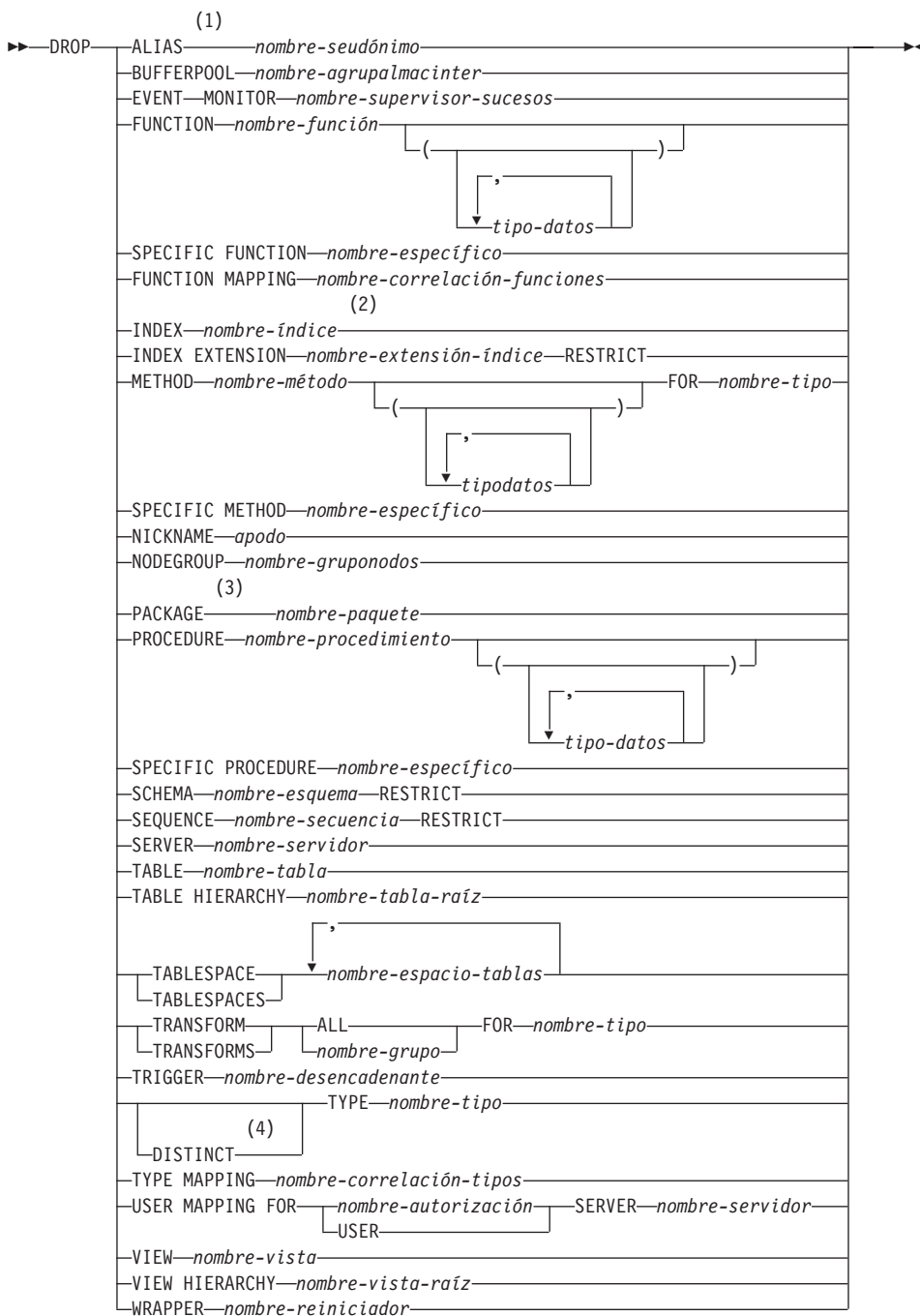
DROP

Cuando se elimina un supervisor de sucesos, una definición de servidor, una correlación de tipos de datos, una función de correlación o reiniciador, el ID de autorización de la sentencia DROP debe tener una autorización SYSADM o DBADM.

Cuando se elimina una correlación de usuarios, intermedios, el ID de autorización de la sentencia DROP debe tener la autorización SYSADM o SYSCTRL si este ID de autorización es diferente del nombre de autorización de la base de datos federada dentro de la correlación. De lo contrario, si el ID de autorización y el nombre de la autorización coinciden, no son obligatorios privilegios o autorizaciones.

Cuando se elimina una transformación, el ID de autorización de la sentencia DROP debe tener la autorización SYSADM o DBADM, o ser el definidor de *nombre-tipo*.

Sintaxis



DROP

Notas:

- 1 SYNONYM puede utilizarse como sinónimo de ALIAS.
- 2 *Nombre-índice* puede ser el nombre de un índice o una especificación de índice.
- 3 PROGRAM puede utilizarse como sinónimo de PACKAGE.
- 4 También puede utilizarse DATA cuando se elimine cualquier tipo definido por el usuario.

Descripción

ALIAS *nombre-seudónimo*

Identifica el seudónimo que se ha de eliminar. El *nombre-seudónimo* debe designar un seudónimo descrito en el catálogo (SQLSTATE 42704). Se suprime el seudónimo especificado.

Todas las tablas, vistas y desencadenantes⁹⁴ que hacen referencia al seudónimo dejan de estar operativas.

BUFFERPOOL *nombre-agrupalmacinter*

Identifica la agrupación de almacenamientos intermedios que se ha de eliminar. El *nombre-agrupalmacinter* debe identificar una agrupación de almacenamientos intermedios que esté descrita en el catálogo (SQLSTATE 42704). Puede que no haya ningún espacio de tablas asignado a la agrupación de almacenamientos intermedios (SQLSTATE 42893). La agrupación de almacenamientos intermedios IBMDEFAULTBP no se puede eliminar (SQLSTATE 42832). El almacenamiento para la agrupación de almacenamientos intermedios no se liberará hasta que se detenga la base de datos.

EVENT MONITOR *nombre-supervisor-sucesos*

Identifica el supervisor de sucesos que se ha de eliminar. El *nombre-supervisor-sucesos* debe identificar un supervisor de sucesos que se describa en el catálogo (SQLSTATE 42704).

Si el supervisor de sucesos identificado es ON, se genera un error (SQLSTATE 55034). De lo contrario, se suprime el supervisor de sucesos.

Si hay archivos de sucesos en la vía de acceso de destino del supervisor de sucesos cuando se elimina el supervisor de sucesos, los archivos de sucesos no se suprimen. Sin embargo, si se crea un nuevo supervisor de sucesos que especifica la misma vía de acceso de destino, entonces se suprimen los archivos de sucesos.

FUNCTION

Identifica una instancia de una función definida por el usuario (una

94. Esto incluye la tabla referenciada en la cláusula ON de la sentencia CREATE TRIGGER y todas las tablas referenciadas en las sentencias de SQL activadas.

función completa o una plantilla de función) que se debe eliminar. La instancia de función especificada debe ser una función definida por el usuario descrita en el catálogo. Las funciones generadas implícitamente por la sentencia CREATE DISTINCT TYPE no se pueden eliminar.

Hay varias maneras diferentes de identificar la instancia de función:

FUNCTION *nombre-función*

Identifica la función específica, y sólo es válido si hay exactamente una instancia de función con el *nombre-función*. La función así identificada puede tener cualquier número de parámetros definidos para la misma. En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objetos no calificados. Si no existe ninguna función con este nombre en el esquema nombrado o implícito, se produce un error (SQLSTATE 42704). Si existe más de una instancia específica de la función en el esquema mencionado o implicado, se producirá un error (SQLSTATE 42725).

FUNCTION *nombre-función (tipo-datos,...)*

Proporciona la signatura de la función, que identifica de manera exclusiva la función que se debe eliminar. El algoritmo de selección de funciones no se utiliza.

nombre-función

Proporciona el nombre de función de la función que se ha de eliminar. En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objetos no calificados.

(tipo-datos, ...)

Debe coincidir con los tipos de datos que se han especificado en la sentencia CREATE FUNCTION en la posición correspondiente. El número de tipos de datos y la concatenación lógica de los tipos de datos se utiliza para identificar la instancia de función específica que se ha de eliminar.

Si *tipo-datos* no está calificado, el nombre de tipo se resuelve efectuando una búsqueda en los esquemas de la vía de acceso de SQL. Esto también se aplica a los nombres de tipo de datos especificados para un tipo REFERENCE.

Para los tipos de datos parametrizados no es necesario especificar la longitud, la precisión ni la escala. En su lugar, se puede codificar un conjunto vacío de paréntesis para indicar que estos atributos se han de ignorar al buscar coincidencias de un tipo de datos.

No puede utilizarse `FLOAT()` (SQLSTATE 42601) puesto que el valor del parámetro indica tipos de datos diferentes (REAL o DOUBLE).

Si se codifica la longitud, la precisión o la escala, el valor debe coincidir exactamente con el especificado en la sentencia `CREATE FUNCTION`.

No es necesario que un tipo de `FLOAT(n)` coincida con el valor definido para `n` puesto que $0 < n < 25$ significa REAL y $24 < n < 54$ significa DOUBLE. La coincidencia se produce basándose en si el tipo es REAL o DOUBLE.

Si no se especifica ninguna función con la signatura especificada en el esquema nombrado o implícito, se genera un error (SQLSTATE 42883).

SPECIFIC FUNCTION *nombre-especifico*

Identifica la función definida por el usuario en particular que se ha de eliminar, utilizando el nombre específico especificado o que toma por omisión en el momento de creación de la función. En las sentencias de SQL dinámicas, el registro especial `CURRENT SCHEMA` se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace `QUALIFIER` especifica implícitamente el calificador para los nombres de objetos no calificados. El *nombre-especifico* debe identificar una instancia de función específica en el esquema nombrado o implícito; de lo contrario, se produce un error (SQLSTATE 42704).

No es posible eliminar una función que esté en el esquema `SYSIBM` o en el esquema `SYSFUN` (SQLSTATE 42832).

Otros objetos pueden depender de una función. Deben eliminarse todas estas dependencias antes de que pueda eliminarse la función, a excepción de los paquetes que están marcados como no operativos. El intento de eliminar una función con dichas dependencias dará como resultado un error (SQLSTATE 42893). Consulte la página 1023 para ver una lista de estas dependencias.

Si la función puede eliminarse, se elimina.

Cualquier paquete dependiente de la función específica que se está eliminando se marca como no operativo. Dicho paquete no se vuelve a

enlazar implícitamente. Debe volverse a enlazar mediante la utilización del mandato BIND o REBIND o debe volverse a preparar mediante la utilización del mandato PREP. Consulte el manual *Consulta de mandatos* para obtener información acerca de estos mandatos.

FUNCTION MAPPING *nombre-correlación-funciones*

Identifica la correlación de funciones que se ha de eliminar. El *nombre-correlación-funciones* debe identificar una correlación de funciones definida por el usuario que esté descrita en el catálogo (SQLSTATE 42704). La correlación de funciones se suprime de la base de datos.

No se pueden suprimir correlaciones de función por omisión. Sin embargo, se pueden inhabilitar. Para ver un ejemplo, consulte el “CREATE FUNCTION MAPPING” en la página 769.

Los paquetes que tengan una dependencia de la correlación de funciones eliminada se invalidarán.

INDEX *nombre-índice*

Identifica el índice o especificación de índice que se ha de eliminar. El *nombre-índice* debe identificar un índice o especificación de índice que se describa en el catálogo (SQLSTATE 42704). No puede ser un índice que necesite el sistema para una restricción de unicidad o clave primaria o para una tabla de resumen duplicada (SQLSTATE 42917). El índice especificado o la especificación de índice se suprime.

Los paquetes que tengan una dependencia de un índice o de una especificación eliminada se invalidarán.

INDEX EXTENSION *nombre-extensión-índice* **RESTRICT**

Identifica la extensión de índice que se debe eliminar. El *nombre-extensión-índice* debe identificar una extensión de índice que esté descrita en el catálogo (SQLSTATE 42704). La palabra clave RESTRICT impide definir cualquier índice que dependa de esta definición de extensión de índice (SQLSTATE 42893).

METHOD

Identifica un cuerpo de método que se debe eliminar. El cuerpo de método especificado debe ser un método descrito en el catálogo (SQLSTATE 42704). Los cuerpos de método que son generados implícitamente por la sentencia CREATE TYPE no se pueden eliminar.

DROP METHOD suprime el cuerpo de un método, pero la especificación del método (signatura) se conserva como parte de la definición del tipo sujeto. Después de eliminar el cuerpo de un método, se puede eliminar la especificación del método en la definición del tipo sujeto, mediante ALTER TYPE DROP METHOD.

Hay varias maneras de especificar el cuerpo de método que se debe eliminar:

METHOD *nombre-método*

Identifica el método en concreto que se debe eliminar y sólo es válido si hay exactamente una instancia de método con el nombre *nombre-método* y el tipo de sujeto *nombre-tipo*. El método así identificado puede tener un número cualquiera de parámetros. Si no existe ningún método con este nombre para el tipo *nombre-tipo*, se produce un error (SQLSTATE 42704). Si existe más de una instancia específica del método para el tipo de datos mencionado, se produce un error (SQLSTATE 42725).

METHOD *nombre-método (tipo-datos,...)*

Proporciona la signatura del método, que identifica de manera exclusiva el método que se debe eliminar. El algoritmo de selección de métodos no se utiliza.

nombre-método

Es el nombre del método que se debe eliminar correspondiente al tipo especificado. El nombre debe ser un identificador no calificado.

(tipo-datos,...)

Debe coincidir con los tipos de datos que se han especificado en la sentencia CREATE FUNCTION o ALTER TYPE, en las posiciones correspondientes la especificación de método. El número de tipos de datos y la concatenación lógica de los tipos de datos se utilizan para identificar la instancia de método específica que se debe eliminar.

Si tipo-datos no está calificado, el nombre de tipo se resuelve efectuando una búsqueda en los esquemas de la vía de acceso de SQL.

Para los tipos de datos parametrizados no es necesario especificar la longitud, la precisión ni la escala. En su lugar, se puede codificar un conjunto vacío de paréntesis para indicar que estos atributos se han de pasar por alto al buscar coincidencias de un tipo de datos.

No puede utilizarse FLOAT() (SQLSTATE 42601) puesto que el valor del parámetro indica tipos de datos diferentes (REAL o DOUBLE).

Sin embargo, si se codifica la longitud, la precisión o la escala, el valor debe coincidir exactamente con el especificado en la sentencia CREATE TYPE.

No es necesario que un tipo de FLOAT(n) coincida con el valor definido para n puesto que $0 < n < 25$ significa REAL y $24 < n < 54$ significa DOUBLE. La coincidencia se produce basándose en si el tipo es REAL o DOUBLE.

Si no existe ningún método que tenga la signatura especificada para el tipo de datos indicado, se produce un error (SQLSTATE 42883).

FOR *nombre-tipo*

Designa el tipo para el cual se debe eliminar el método especificado. El nombre debe identificar un tipo que ya esté descrito en el catálogo (SQLSTATE 42704). En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de tipo no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de tipo no calificados.

SPECIFIC METHOD *nombre-específico*

Identifica el método específico que se debe eliminar, mediante un nombre especificado o un nombre que se toma por omisión al ejecutar CREATE TYPE o ALTER TYPE. En el SQL dinámico, si el nombre específico no está calificado, se utiliza el registro especial CURRENT SCHEMA como calificador para un nombre específico no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para un nombre específico no calificado. El nombre específico debe identificar un método; de lo contrario se produce un error (SQLSTATE 42704).

Otros objetos pueden depender de un método. Todas estas dependencias deben eliminarse para poder eliminar el método, con la excepción de los paquetes, que se marcarán como no operativos si la eliminación es efectiva. El intento de eliminar un método con dichas dependencias dará como resultado un error (SQLSTATE 42893).

Si el método se puede eliminar, se eliminará.

Los paquetes dependientes del método específico que se está eliminando se marcarán como no operativos. Dichos paquetes no se vuelven a enlazar implícitamente. Deben volverse a enlazar mediante el mandato BIND o REBIND, o deben volverse a preparar mediante el mandato PREP. Vea el manual *Consulta de mandatos* para obtener información sobre estos mandatos.

NICKNAME *apodo*

Identifica el apodo que se ha de eliminar. El apodo debe estar listado en el catálogo (SQLSTATE 42704). El apodo se suprime de la base de datos.

Toda la información acerca de las columnas e índices asociados con el apodo se elimina del catálogo. Se elimina cualquier especificación de índice que es dependiente del apodo. Cualquier vista dependiente del apodo se marca como no operativa. Cualquier paquete dependiente de

DROP

especificaciones de índice o de vistas no operativas eliminadas es invalidado. No afecta a la tabla de fuente de datos al que el apodo hace referencia.

NODEGROUP *nombre-gruponodos*

Identifica el grupo de nodos que se ha de eliminar. El *nombre-gruponodos* debe identificar un grupo de nodos que esté descrito en el catálogo (SQLSTATE 42704). Este nombre consta de una sola parte.

La eliminación de un grupo de nodos elimina todos los espacios de tablas definidos en el grupo de nodos. Todos los objetos de base de datos existentes con dependencias en las tablas de los espacios de tablas (como paquetes, restricciones de referencia, etc.) se eliminan o invalidan (según sea lo adecuado) y las vistas y desencadenantes dependientes pasan a ser no operativos.

Los grupos de nodos definidos por el sistema no pueden eliminarse (SQLSTATE 42832).

Si se emite DROP NODEGROUP para un grupo de nodos que está sometido actualmente a una redistribución de datos, la operación DROP NODEGROUP falla y se devuelve un error (SQLSTATE 55038). Sin embargo, un grupo de nodos redistribuido parcialmente puede eliminarse. Un grupo de nodos puede redistribuirse parcialmente si un mandato REDISTRIBUTE NODEGROUP no se ejecuta hasta completarse. Esto puede ocurrir si se interrumpe por un error o mandato.⁹⁵

PACKAGE *nombre-paquete*

Identifica el paquete que se ha de eliminar. El *nombre-paquete* debe identificar un paquete que esté descrito en el catálogo (SQLSTATE 42704). Se suprime el paquete especificado. También se suprimen todos los privilegios del paquete.

PROCEDURE

Identifica una instancia de un procedimiento almacenado que no se puede eliminar. La instancia del procedimiento especificado debe ser un procedimiento almacenado descrito en el catálogo.

Hay varias maneras diferentes de identificar la instancia de procedimiento:

PROCEDURE *nombre-procedimiento*

Identifica el procedimiento en particular que se debe eliminar y sólo es válido si hay exactamente una instancia de procedimiento con el *nombre-procedimiento* en el esquema. El procedimiento identificado de esta manera puede tener cualquier número de parámetros definido. Si no existe ningún procedimiento con este nombre en el esquema

95. Para un grupo de nodos redistribuido parcialmente, REBALANCE_PMAP_ID en el catálogo SYSCAT.NODEGROUPS no es -1.

nombrado o implícito, se genera un error (SQLSTATE 42704). En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objetos no calificados. Si existe más de una instancia específica del procedimiento en el esquema mencionado o implicado, se producirá un error (SQLSTATE 42725).

PROCEDURE *nombre-procedimiento (tipo-datos,...)*

Proporciona la signatura del método, que identifica de manera exclusiva el procedimiento que se debe eliminar. El algoritmo de selección de procedimientos no se utiliza.

nombre-procedimiento

Proporciona el nombre del procedimiento que se debe eliminar. En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objetos no calificados.

(tipo-datos, ...)

Deben coincidir con los tipos de datos que se han especificado en la sentencia CREATE PROCEDURE en la posición correspondiente. El número de tipos de datos y la concatenación lógica de los tipos de datos se utilizan para identificar la instancia específica del procedimiento que se ha de eliminar.

Si *tipo-datos* no está calificado, el nombre de tipo se resuelve efectuando una búsqueda en los esquemas de la vía de acceso de SQL. Esto también se aplica a los nombres de tipo de datos especificados para un tipo REFERENCE.

Para los tipos de datos parametrizados no es necesario especificar la longitud, la precisión ni la escala. En su lugar, se puede codificar un conjunto vacío de paréntesis para indicar que estos atributos se han de ignorar al buscar coincidencias de un tipo de datos.

No puede utilizarse FLOAT() (SQLSTATE 42601) puesto que el valor del parámetro indica tipos de datos diferentes (REAL o DOUBLE).

De todas formas, si la longitud, la precisión o la escala están codificadas, el valor debe coincidir exactamente con el que se especifica en la sentencia CREATE FUNCTION.

No es necesario que un tipo de FLOAT(n) coincida con el valor definido para n puesto que $0 < n < 25$ significa REAL y $24 < n < 54$ significa DOUBLE. La coincidencia se produce basándose en si el tipo es REAL o DOUBLE.

Si no existe ningún procedimiento con la signatura especificada en el esquema nombrado o implícito, se genera un error (SQLSTATE 42883).

SPECIFIC PROCEDURE *nombre-específico*

Identifica el procedimiento almacenado en particular que se ha de eliminar, utilizando el nombre específico que se ha especificado o que ha tomado por omisión en el momento de la creación del procedimiento. En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objetos no calificados. El *nombre-específico* debe identificar una instancia del procedimiento específico en el esquema nombrado o implícito; de lo contrario, se genera un error (SQLSTATE 42704).

SCHEMA *nombre-esquema* **RESTRICT**

Identifica el esquema en particular que se debe eliminar. El *nombre-esquema* debe identificar un esquema que está descrito en el catálogo (SQLSTATE 42704). La palabra clave RESTRICT impone la regla de que no puede haber ningún objeto definido en el esquema especificado para que se suprima de la base de datos (SQLSTATE 42893).

SEQUENCE *nombre-secuencia* **RESTRICT**

Identifica la secuencia en particular que se debe eliminar. El *nombre-secuencia*, junto con el nombre de esquema implícito o explícito, debe identificar una secuencia existente en el servidor actual. Si no existe ninguna secuencia con este nombre en el esquema especificado explícita o implícitamente, se produce un error (SQLSTATE 42704).

La palabra clave RESTRICT impide que se elimine la secuencia si existe alguna de las dependencias siguientes:

- Existe un desencadenante de tal modo que una expresión NEXTVAL o PREVVAL en el desencadenante especifica la secuencia (SQLSTATE 42893).
- Existe una rutina SQL incorporada de tal modo que una expresión NEXTVAL o PREVVAL en el cuerpo de la rutina especifica la secuencia (SQLSTATE 42893).

SERVER *nombre-servidor*

Identifica la fuente de datos cuya definición se ha de eliminar del

catálogo. El *nombre-servidor* debe identificar una fuente de datos que está descrita en el catálogo (SQLSTATE 42704). Se elimina la definición de la fuente de datos.

Se eliminan todos los apodos para tablas y vistas que residen en la fuente de datos. Se elimina cualquier especificación de índice dependiente de estos apodos. También es eliminada cualquier correlación de funciones definida por el usuario, correlación de tipos definida por el usuario y correlación de usuarios que es dependiente de la definición de servidor. Se invalidan todos los paquetes dependientes de la definición de servidor, correlaciones de función, apodos y especificaciones de índices eliminados.

TABLE *nombre-tabla*

Identifica la tabla base, tabla temporal declarada o tabla de resumen que se debe eliminar. El *nombre-tabla* debe identificar una tabla que está descrita en el catálogo o, si es una tabla temporal declarada, el *nombre-tabla* debe estar calificado por el nombre de esquema SESSION y existir en la aplicación (SQLSTATE 42704). Las subtablas de una tabla con tipo dependen de sus supertablas. Deben eliminarse todas las subtablas antes de poder eliminar una supertabla (SQLSTATE 42893). La tabla especificada se suprime de la base de datos.

Se eliminan todos los índices, claves primarias, claves foráneas, restricciones de comprobación y tablas de resumen que hagan referencia a la tabla. Todas las vistas y desencadenantes ⁹⁶ que hacen referencia a la tabla pasan a estar no operativos. Todos los paquetes que dependen de cualquier objeto eliminado o marcado como no operativo se invalidarán. Esto incluye los paquetes que dependan de cualquier supertabla por encima de la subtabla en la jerarquía. Las columnas de referencia para las que la tabla eliminada se defina como ámbito de la referencia se quedan sin ámbito.

Los paquetes no dependen de tablas temporales declaradas, y por tanto no se invalidan cuando se elimina una tabla esa clase.

Todos los archivos que estén enlazados mediante columnas DATALINK se desenlazan. La operación de desenlace se lleva a cabo de manera asíncrona, por lo que es posible que los archivos no estén inmediatamente disponibles para otras operaciones.

Cuando se elimina una subtabla de una jerarquía de tablas, las columnas asociadas con la subtabla ya no son accesibles aunque sigan teniéndose en cuenta con respecto a los límites del número de columnas y tamaño de la fila. Cuando se elimina una subtabla, todas sus filas se suprimen en las supertablas. Ello puede provocar la activación de desencadenantes o de restricciones de integridad referencial definidos para las supertablas.

⁹⁶. Esto incluye tanto la tabla a la que la cláusula ON de la sentencia CREATE TRIGGER hace referencia como todas las tablas a las que se hace referencia en las sentencias de SQL activadas.

Cuando se elimina una tabla temporal declarada, y su creación fue anterior a la unidad de trabajo activa o punto de salvar, se inhabilita la tabla y la aplicación no puede acceder a ella. Sin embargo, la tabla todavía conservará algo de espacio en su espacio de tablas y evitará que se elimine el espacio de tablas USER TEMPORARY o que se redistribuya el grupo de nodos del espacio de tablas USER TEMPORARY hasta que se confirme la unidad de trabajo o finalice el punto de salvar. La eliminación de una tabla temporal declarada hace que se destruyan los datos de la tabla, con independencia de si DROP se confirma o retrotrae.

TABLE HIERARCHY *nombre-tabla-raíz*

Identifica la jerarquía de la tabla con tipo que se ha de eliminar. El *nombre-tabla-raíz* debe identificar una tabla con tipo que es tabla raíz de la jerarquía de tablas con tipo (SQLSTATE 428DR). La tabla con tipo identificada mediante el *nombre-tabla-raíz* y todas sus subtablas se suprimen de la base de datos.

Se eliminan todos los índices, tablas de resumen, claves primarias, claves foráneas y restricciones de comprobación que hacen referencia a las tablas eliminadas. Todas las vistas y desencadenantes que hacen referencia a las tablas eliminadas se hacen no operativas. Todos los paquetes que dependen de cualquier objeto eliminado o marcado como no operativo se invalidarán. Todas las columnas de referencia para las que una de las tablas eliminadas se define como ámbito de la referencia se quedan sin ámbito.

Todos los archivos que estén enlazados mediante columnas DATALINK se desenlazan. La operación de desenlace se lleva a cabo de manera asíncrona, por lo que es posible que los archivos no estén inmediatamente disponibles para otras operaciones.

A diferencia de la eliminación de una subtabla individual, la eliminación de la jerarquía de tablas no produce la activación de los desencadenantes de supresión en ninguna tabla de la jerarquía ni registra en el archivo de anotaciones las filas suprimidas.

TABLESPACE o **TABLESPACES** *nombre-espacio-tablas*

Identifica los espacios de tablas que se han de eliminar. El *nombre-espacio-tablas* debe identificar un espacio tabla que se describa en el catálogo (SQLSTATE 42704). Este nombre consta de una sola parte.

Los espacios de tablas no se eliminarán (SQLSTATE 55024) si existe alguna tabla que guarda al menos una de sus partes en un espacio de tablas que se está eliminando y tiene una o más de sus partes en otro espacio de tablas que no se está eliminando (sería necesario eliminar primero estas tablas). Los espacios de tablas del sistema no se pueden eliminar (SQLSTATE 42832). Un espacio de tablas temporal del sistema (SYSTEM TEMPORARY) no se puede eliminar (SQLSTATE 55026) si es el único espacio de tablas temporal que existe en la base de datos. Un espacio de

tablas temporal del usuario (USER TEMPORARY) no se puede eliminar si en él existe una tabla temporal declarada (SQLSTATE 55039). Aunque se haya eliminado una tabla temporal declarada, se considera que el espacio de tablas USER TEMPORARY todavía está en uso hasta que se confirme la unidad de trabajo de la sentencia DROP TABLE .

La eliminación de un espacio de tablas elimina todos los objetos definidos en el espacio de tablas. Todos los objetos de base de datos existentes con dependencias en el espacio de tablas como, por ejemplo, paquetes, restricciones de referencia, etc. se eliminan o invalidan (lo que sea adecuado) y las vistas y desencadenantes dependientes pasan a estar no operativos.

No se suprimen los contenedores creados por el usuario no. Se suprimirá cualquier directorio en la vía de acceso del nombre de contenedor que se haya creado por el gestor de bases de datos en CREATE TABLESPACE. Se suprimen todos los contenedores que están por debajo del directorio de bases de datos. Para los espacios de tablas SMS, las supresiones se producen después de desconectar todas las conexiones o de emitir el mandato DEACTIVATE DATABASE.

TRANSFORM ALL FOR *nombre-tipo*

Indica que se deben eliminar todos los grupos de transformación definidos para el tipo de datos definido por el usuario, *nombre-tipo*. Las funciones de transformación referenciadas en estos grupos no se eliminan. En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objeto no calificados. El *nombre-tipo* debe identificar un tipo definido por el usuario que esté descrito en el catálogo (SQLSTATE 42704).

Si no hay transformaciones definidas para *nombre-tipo*, se emite un error (SQLSTATE 42740).

DROP TRANSFORM es lo opuesto de CREATE TRANSFORM. Hace que las funciones de transformación asociadas a determinados grupos, para un tipo de datos proporcionado, pasen a estar no definidas. Las funciones que estaban asociadas a estos grupos siguen existiendo y todavía pueden invocarse explícitamente, pero ya no tienen el atributo de transformación y no se invocan implícitamente para intercambiar valores con el entorno del lenguaje principal.

El grupo de transformación no se elimina si existe una función (o método) definida por el usuario, escrita en un lenguaje distinto del SQL, que depende de una de las funciones de transformación del grupo definidas para el tipo definido por el usuario *nombre-tipo* (SQLSTATE 42893). Dicha función depende de la función de transformación asociada al grupo de

transformación referenciado que se ha definido para el tipo *nombre-tipo*. Los paquetes que dependen de una función de transformación asociada al grupo de transformación referenciado se marcan como no operativos.

TRANSFORMS *nombre-grupo* **FOR** *nombre-tipo*

Indica que debe eliminarse el grupo de transformación especificado para el tipo de datos definido por el usuario *nombre-tipo*. Las funciones de transformación referenciadas en este grupo no se eliminan. En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objeto no calificados. El *nombre-tipo* debe identificar un tipo definido por el usuario que esté descrito en el catálogo (SQLSTATE 42704), y el *nombre-grupo* debe identificar un grupo de transformación existente para *nombre-tipo*.

TRIGGER *nombre-desencadenante*

Identifica el desencadenante que se ha de eliminar. El *nombre-desencadenante* debe identificar un desencadenante que esté descrito en el catálogo (SQLSTATE 42704). Se suprime el desencadenante especificado.

La eliminación de desencadenantes hace que algunos paquetes se marquen como no válidos. Consulte la sección “Notas” del apartado “CREATE TRIGGER” en la página 910 en relación a la creación de desencadenantes (que sigue las mismas reglas).

TYPE *nombre-tipo*

Identifica el tipo definido por el usuario que se ha de eliminar. En las sentencias de SQL dinámicas, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estáticas, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objetos no calificados. Para un tipo estructurado, también se elimina el tipo de referencia asociado. El *nombre-tipo* debe identificar un tipo definido por el usuario descrito en el catálogo. Si se especifica DISTINCT, entonces el *nombre-tipo* debe identificar un tipo diferenciado que esté descrito en el catálogo. El tipo no se elimina (SQLSTATE 42893) si es verdadero cualquiera de los puntos siguientes.

- El tipo se utiliza como tipo de una columna de una tabla o vista.
- El tipo tiene un subtipo.
- El tipo es un tipo estructurado que se utiliza como tipo de datos de una tabla con tipo o vista con tipo.
- El tipo es un atributo de otro tipo estructurado.
- Existe una columna de una tabla cuyo tipo puede contener una instancia de *nombre-tipo*. Esto puede ocurrir si *nombre-tipo* es el tipo de

la columna o se utiliza en otro lugar de la jerarquía de tipos asociada de la columna. Es decir, para cualquier tipo T, no se puede eliminar T si existe una columna de una tabla cuyo tipo utiliza, directa o indirectamente, *nombre-tipo*.

- El tipo es el tipo destino de una columna de tipo de referencia de la tabla o vista o un atributo de tipo de referencia de otro tipo estructurado.
- El tipo o una referencia al tipo es un tipo de parámetro o un tipo de valor de retorno de una función o método que no puede eliminarse.
- El tipo, o una referencia al tipo, se utiliza en el cuerpo de una función o método SQL, pero no es un tipo de parámetro ni un tipo de valor de retorno.
- El tipo se utiliza en una restricción de comprobación, un desencadenante, una definición de vista o en una extensión de índice.

Funciones que utilizan el tipo: Si puede eliminarse el tipo definido por el usuario, para cada función F (con el nombre específico SF), que tenga parámetros o un valor de retorno del tipo que se elimina, o una referencia al tipo que se elimina, la siguiente sentencia DROP FUNCTION se ejecuta eficazmente:

```
DROP SPECIFIC FUNCTION SF
```

Es posible que esta sentencia también elimine en cascada las funciones dependientes. Si todas estas funciones también están en la lista que se ha de eliminar debido a una dependencia del tipo definido por el usuario, la eliminación del tipo definido por el usuario será satisfactoria (de lo contrario, falla con SQLSTATE 42893).

Métodos que utilizan el tipo: Si puede eliminarse el tipo definido por el usuario, para cada método M de tipo T1 (con el nombre específico SM), que tenga parámetros o un valor de retorno del tipo que se elimina, o una referencia al tipo que se elimina, las siguientes sentencias se ejecutan eficazmente:

```
DROP SPECIFIC METHOD SM  
ALTER TYPE T1 DROP SPECIFIC METHOD SM
```

La existencia de objetos que dependen de estos métodos puede hacer que la sentencia DROP TYPE no sea efectiva.

TYPE MAPPING *nombre-correlación-tipos*

Identifica la correlación de tipos de datos definida por el usuario que se ha de eliminar. El *nombre-correlación-tipos* debe identificar una correlación de tipos de datos que esté descrita en el catálogo (SQLSTATE 42704). La correlación de tipos de datos se suprime de la base de datos.

No se eliminan objetos adicionales.

USER MAPPING FOR *nombre-autorización* | **USER SERVER** *nombre-servidor*

Identifica la correlación de usuarios que se ha de eliminar. Esta correlación asocia un nombre de autorización que se utiliza para acceder a la base de datos federada con un nombre de autorización que se utiliza para acceder a la fuente de datos. Se identifica el primero de estos dos nombres de autorización mediante el *nombre-autorización* o se hace referencia mediante el registro especial USER. El *nombre-servidor* identifica la fuente de datos que el segundo nombre de autorización utiliza para tener acceso.

El *nombre-autorización* debe estar listado en el catálogo (SQLSTATE 42704). El *nombre-servidor* debe identificar una fuente de datos que está descrita en el catálogo (SQLSTATE 42704). Se suprime la correlación del usuario.

No se eliminan objetos adicionales.

VIEW *nombre-vista*

Identifica la vista que se ha de eliminar. El *nombre-vista* debe identificar una vista que esté descrita en el catálogo (SQLSTATE 42704). Las subvistas de una vista con tipo dependen de sus supervistas. Deben eliminarse todas las subvistas antes de poder eliminar una supervista (SQLSTATE 42893).

Se suprime la vista especificada. La definición de cualquier vista o desencadenante que sea dependiente directa o indirectamente de esta vista se marca como no operativa. Se eliminan las tablas de resumen que dependan de cualquier vista marcada como no operativa. Se invalidará cualquier paquete que dependa de una vista que se elimine o se marque como no operativa. Esto incluye los paquetes que dependan de cualquier supervista por encima de la subvista en la jerarquía. Las columnas de referencia para las que la vista eliminada se defina como ámbito de la referencia se quedan sin ámbito.

VIEW HIERARCHY *nombre-vista-raíz*

Identifica la jerarquía de vistas con tipo que se debe eliminar. El *nombre-vista-raíz* debe identificar una vista con tipo que es vista raíz de la jerarquía de vistas con tipo (SQLSTATE 428DR). Se suprimen de la base de datos la vista con tipo identificada mediante el *nombre-vista-raíz* y todas sus subvistas.

La definición de cualquier vista o desencadenante que sea directa o indirectamente dependiente de cualquiera de las vista eliminadas se marca como no operativa. Cualquier paquete que dependa de cualquier vista o desencadenante que se elimine o se marque como no operativo será invalido. Cualquier columna de referencia para la que una vista eliminada o vista marcada como no operativa se define como ámbito de la referencia se queda sin ámbito.

WRAPPER *nombre-reiniciador*

Identifica el reiniciador que se debe eliminar. El *nombre-reiniciador* debe identificar un reiniciador que esté descrito en el catálogo (SQLSTATE 42704). Se suprime el reiniciador.

Se eliminan todas las definiciones de servidor, correlaciones de función definidas por el usuario y correlaciones de tipo de datos definidas por el usuario que dependen del reiniciador. También se eliminan todas las correlaciones de función definidas por el usuario, apodos, correlaciones de tipo de datos definidas por el usuario y correlaciones de usuario que son dependientes de las definiciones de servidor. Se elimina cualquier especificación de índice dependiente de estos apodos eliminados y se marca como no operativa cualquier vista dependiente de estos apodos. Se invalidan todos los paquetes dependientes de los objetos eliminados y vistas no operativas.

Normas

Dependencias: La Tabla 29 en la página 1024 muestra las dependencias que los objetos tienen entre sí.⁹⁷Se muestran cuatro tipos de dependencias distintas:

- R** Semántica de restricción. El objeto principal no puede eliminarse mientras exista el objeto que depende de él.
- C** Semántica en cascada. La eliminación del objeto principal hace que el objeto que depende de él (objeto dependiente) se elimine también. Sin embargo, si el objeto dependiente no puede eliminarse debido a que tiene una dependencia de restricción en otro objeto, la eliminación del objeto principal fallará.
- X** Semántica de no operativo. La eliminación del objeto principal hace que el objeto que depende de él pase a estar no operativo. Permanece no operativo hasta que un usuario lleva a cabo una acción explícita.
- A** Semántica de invalidación/revalidación automática. La eliminación del objeto principal hace que el objeto que depende del mismo pase a ser no válido. El gestor de bases de datos intenta revalidar el objeto no válido.

Algunos objetos y parámetros de la sentencia DROP no se muestran en Tabla 29 en la página 1024 porque darían como resultado columnas o filas en blanco:

- Las sentencias EVENT MONITOR, PACKAGE, PROCEDURE, SCHEMA, TYPE MAPPING y USER MAPPING DROP no tienen dependencias de objeto.

⁹⁷. No todas las dependencias se registran explícitamente en el catálogo. Por ejemplo, no hay ningún registro de las restricciones de las que depende un paquete.

DROP

- No tienen dependencias de sentencia DROP los seudónimos, agrupaciones de almacenamiento intermedio, claves de particionamiento, privilegios y tipos de objetos de procedimiento.
- Una sentencia DROP SERVER, DROP FUNCTION MAPPING, o DROP TYPE MAPPING en una unidad de trabajo dada (UOW) no puede procesarse bajo ninguna de las condiciones siguientes:
 - La sentencia hace referencia a una sola fuente de datos y la UOW ya incluye una sentencia SELELCT que hace referencia a un apodo para una tabla o vista dentro de esta fuente de datos (SQLSTATE 55006).
 - La sentencia hace referencia a una categoría de una fuente de datos (por ejemplo, todas las fuentes de datos para un tipo específico y versión) y la UOW ya incluye una sentencia SELELCT que hace referencia a un apodo para una tabla o vista dentro de estas fuentes de datos (SQLSTATE 55006).

Tabla 29. Dependencias

Tipo de objeto →	C O N S T R A I N	F U N C T I O N S	I N D E X	N O T	M E M O R Y	P R O C E D U R E S	S E R V E R S	T A B L E S	T R I G G E R S	T Y P E S	U S E R D E F I N E D
ALTER NICKNAME	-	-	-	-	-	-	-	A	-	-	-
ALTER SERVER	-	-	-	-	-	-	-	A	-	-	-
ALTER TABLE DROP CONSTRAINT	C	-	-	-	-	-	-	A ¹	-	-	-
ALTER TABLE DROP PARTITIONING KEY	-	-	-	-	-	-	R ²⁰	A ¹	-	-	-
ALTER TYPE ADD ATTRIBUTE	-	-	-	-	R	-	-	A ²³	-	R ²⁴	-
ALTER TYPE DROP ATTRIBUTE	-	-	-	-	R	-	-	A ²³	-	R ²⁴	-

Tabla 29. Dependencias (continuación)

Tipo de objeto →	C O N S T R A I N	F U N C T I O N	M A P I N G	I N D E X	E X T E N S I O N	M E T H O D	N I C K N A M E	N O D E G R O U P	P A R T I C I P A N T	S E R I A L	T A B L E S	T R I G G E R	T Y P E	S E R V E R	U S E R	V I E W	
Sentencia ↓	T	N	G	X	N	D	E	P	E	R	E	R	E	G	N	G	W
ALTER TYPE ADD METHOD	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ALTER TYPE DROP METHOD	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DROP ALIAS	-	R	-	-	-	-	-	-	A ³	-	R ³	-	X ³	-	-	-	X ³
DROP BUFFERPOOL	-	-	-	-	-	-	-	-	-	-	R	-	-	-	-	-	-
DROP FUNCTION	R	R ⁷	R	-	R	R ⁷	-	-	X	-	R	-	R	-	-	-	R
DROP FUNCTION MAPPING	-	-	-	-	-	-	-	-	A	-	-	-	-	-	-	-	-
DROP INDEX	R	-	-	-	-	-	-	-	A	-	-	-	-	-	-	-	R ¹⁷
DROP INDEX EXTENSION	-	R	-	R	-	-	-	-	-	-	-	-	-	-	-	-	-
DROP METHOD	R	R ⁷	R	-	R	R	-	-	X	-	R	-	R	-	-	-	R
DROP NICKNAME	-	R	-	C	-	-	-	-	A	-	-	-	-	-	-	-	X ¹⁶
DROP NODEGROUP	-	-	-	-	-	-	-	-	-	-	-	C	-	-	-	-	-
DROP SEQUENCE	-	-	-	-	-	-	-	-	A	-	-	-	R	-	-	-	-
DROP SERVER	-	C ²¹	C ¹⁹	-	-	-	C	-	A	-	-	-	-	-	C ¹⁹	C	-
DROP TABLE	C	R	-	C	-	-	-	-	A ⁹	-	RC ¹¹	-	X ¹⁶	-	-	-	X ¹⁶
DROP TABLE HIERARCHY	C	R	-	C	-	-	-	-	A ⁹	-	RC ¹¹	-	X ¹⁶	-	-	-	X ¹⁶
DROP TABLESPACE	-	-	-	C ⁶	-	-	-	-	-	-	CR ⁶	-	-	-	-	-	-
DROP TRANSFORM	-	R	-	-	-	-	-	-	X	-	-	-	-	-	-	-	-
DROP TRIGGER	-	-	-	-	-	-	-	-	A ¹	-	-	-	-	-	-	-	-

DROP

Tabla 29. Dependencias (continuación)

Tipo de objeto →	C O N S T R A I N T	F U N C I O N S	M A T R I C E S	I N D E X E S	D E F I N I C I O N E S	M E M O R I A S	N O T I C I A S	O B J E T O S	P A C K E T E S	S E R I E S	T I P O S	T A B L A S	T R I G G E R S	T I P O S	T I P O S	T I P O S	T I P O S	T I P O S
DROP TYPE	R ¹³	R ⁵	-	-	R	-	-	-	A ¹²	-	R ¹⁸	-	R ¹³	R ⁴	-	-	-	R ¹⁴
DROP VIEW	-	R	-	-	-	-	-	-	A ²	-	-	-	X ¹⁶	-	-	-	-	X ¹⁵
DROP VIEW HIERARCHY	-	R	-	-	-	-	-	-	A ²	-	-	-	X ¹⁶	-	-	-	-	X ¹⁶
DROP WRAPPER	-	-	C	-	-	-	-	-	-	C	-	-	-	-	-	C	-	-
REVOKE un privilegio ¹⁰	-	CR ²⁵	-	-	-	-	-	-	A ¹	-	CX ⁸	-	X	-	-	-	-	X ⁸

- 1 Esta dependencia está implícita en la dependencia de una tabla con estas restricciones, desencadenantes o clave de particionamiento.
- 2 Si un paquete tiene una sentencia INSERT, UPDATE o DELETE que actúa en una vista, el paquete tiene una utilización de inserción, actualización o supresión en la tabla base principal de la vista. En el caso de UPDATE, el paquete tiene una utilización de actualización en cada columna de la tabla base principal que se modifica por UPDATE.

Si un paquete tiene una sentencia que actúa en una vista con tipo, la creación o eliminación de cualquier vista de la misma jerarquía de vistas invalidará el paquete.
- 3 Si un paquete, tabla de resumen, vista o desencadenante utiliza un seudónimo, se convierte en dependiente del seudónimo y del objeto al que el seudónimo hace referencia. Si el seudónimo está en una cadena, se crea una dependencia de cada seudónimo de la cadena.

Los propios seudónimos no son dependientes de nada. Es posible definir un seudónimo de un objeto que no exista.
- 4 Un tipo T definido por el usuario puede depender de otro tipo B definido por el usuario si T:
 - designa B como tipo de datos de un atributo
 - tiene un atributo de REF(B)

- tiene B como supertipo.
- 5 La eliminación de un tipo de datos se propaga y elimina las funciones y métodos que hacen uso de ese tipo de datos como parámetro o tipo resultante, y los métodos definidos para el tipo de datos. La eliminación de estas funciones y métodos no se ve impedida por el hecho de sean dependientes entre sí. Sin embargo, para las funciones o métodos que utilizan el tipo de datos dentro de sus cuerpos, se aplican semánticas de restricción.
 - 6 Eliminar un espacio de tablas o una lista de espacios de tablas hace que se eliminen todas las tablas que están totalmente contenidas dentro de este espaciotablas o lista. Sin embargo, si una tabla se fragmenta en espacios de tablas (los índices o columnas largas en diferentes espacios de tablas) y esos espacios de tablas no están en la lista que se ha eliminado, entonces no se puede eliminar ninguno de estos espacios de tablas mientras la tabla exista.
 - 7 Una función puede depender de otra función específica si la función dependiente nombra la función de base en una cláusula SOURCE. Una función o método puede también depender de otra función o método determinados si la rutina dependiente está escrita en SQL y utiliza la rutina base en su cuerpo. Un método externo o una función externa con un parámetro de tipo estructurado o tipo de retorno dependerá también de una o más funciones de transformación.
 - 8 Sólo la pérdida del privilegio SELECT hará que se elimine una tabla de resumen o se inhabilite una vista. Si la vista que se inhabilita está incluida en una jerarquía de vistas con tipo, también se inhabilitarán todas sus subvistas.
 - 9 Si un paquete tiene una sentencia INSERT, UPDATE o DELETE que actúa en una tabla T, el paquete tiene un uso de inserción, actualización o supresión en T. En el caso de UPDATE, el paquete tiene un uso de actualización en cada columna de T que se modifica por UPDATE.

Si un paquete tiene una sentencia que actúa en una tabla con tipo, la creación o eliminación de cualquier tabla de la misma jerarquía de tablas invalidará el paquete.
 - 10 Las dependencias no existen en el nivel de columna porque los privilegios en las columnas no se pueden revocar individualmente.

Si un paquete, un desencadenante o una vista incluye la utilización de OUTER(Z) en la cláusula FROM, existe una dependencia en el privilegio SELECT en cada subtabla o subvista de Z. De modo similar, si un paquete, un desencadenante o una vista incluye la utilización de

DROP

DEREF(*Y*) donde *Y* es un tipo de referencia con una vista o tabla de destino *Z*, hay una dependencia del privilegio SELECT sobre cada subtabla o subvista de *Z*.

- 11 Una tabla de resumen depende de la tabla o tablas principales especificadas en la selección completa de la definición de tabla.

La semántica de la propagación en cascada es aplicable a las tablas de resumen dependientes.

Una subtabla depende de sus supertablas hasta la tabla raíz. No se puede eliminar una supertabla hasta que se hayan eliminado todas sus subtablas.
- 12 Un paquete puede depender de tipos estructurados como resultado de utilizar el predicado TYPE o la expresión de tratamiento de subtipos (TREAT*expresión AS tipo-datos*). El paquete depende de los subtipos de cada tipo estructurado especificado en el lado derecho del predicado TYPE o de la expresión TREAT. La eliminación o creación de un tipo estructurado que altera los subtipos de los que el paquete depende causa la invalidación.
- 13 Una restricción de comprobación o un desencadenante depende de un tipo si el tipo se utiliza en cualquier parte dentro de la restricción o del desencadenante. No hay dependencia de los subtipos de un tipo estructurado utilizado en un predicado TYPE dentro de una restricción de comprobación o un desencadenante.
- 14 Una vista depende de un tipo si el tipo se utiliza en cualquier parte dentro de la definición de vista (esto incluye el tipo de la vista con tipo). No hay dependencia de los subtipos de un tipo estructurado utilizado en un predicado TYPE dentro de una definición de vista.
- 15 Una subvista depende de su supervista hasta la vista raíz. No se puede eliminar una supervista hasta que se hayan eliminado todas sus subvistas. Consulte el número ¹⁶ para obtener dependencias de vista adicionales.
- 16 Un desencadenante o una vista también depende de la tabla de destino o vista de destino de una operación de desreferencia o función Deref. Un desencadenante o una vista con una cláusula FROM que incluya OUTER(*Z*) depende de todas las subtablas o subvistas de *Z* que existían en el momento de crearse el desencadenante o la vista.
- 17 Una vista con tipo puede depender de la existencia de un índice de unicidad para asegurar la exclusividad de la columna de identificador de objeto.
- 18 Una tabla puede depender de un tipo de datos definido por el usuario (diferenciado o estructurado) porque:
 - el tipo se utiliza como tipo de una columna

- el tipo se utiliza como tipo de la tabla
 - el tipo se utiliza como atributo de un tipo de la tabla
 - el tipo se utiliza como tipo destino de un tipo de referencia que es el tipo de una columna de la tabla o un atributo del tipo de la tabla
 - el tipo es utilizado, directa o indirectamente, por un tipo que es la columna de la tabla.
- 19 La eliminación de un servidor produce la eliminación en cascada de las correlaciones de funciones y tipo de correlaciones creadas para ese servidor nombrado.
- 20 Si la clave de particionamiento se define en una tabla en un nodogrupa de partición múltiple, es obligatoria la clave de particionamiento.
- 21 Si una función de tabla dependiente OLE DB tiene "R" objetos dependientes (véase DROP FUNCTION), el servidor no se puede eliminar.
- 22 Una función o método SQL puede depender de los objetos referenciados por su cuerpo.
- 23 Cuando se elimina un atributo A de tipo TA de *nombre-tipo* T, las sentencias DROP siguientes son efectivas:
- ```
Método mutador: DROP METHOD A (TA) FOR T
Método observador: DROP METHOD A () FOR T
ALTER TYPE T
 DROP METHOD A(TA)
 DROP METHOD A()
```
- 24 Una tabla puede depender de un atributo de un tipo de datos estructurado definido por el usuario en los casos siguientes:
1. La tabla es una tabla con tipo que está basada en *nombre-tipo* o en cualquiera de sus subtipos.
  2. La tabla tiene una columna de un tipo que, directa o indirectamente, hace referencia a *nombre-tipo*.
- 25 La revocación de un privilegio SELECT para una tabla o vista que se utiliza en el cuerpo de una función SQL provoca un intento de eliminar la función, si la función definida ya no tiene el privilegio SELECT WITH GRANT OPTION. Si dicha función se utiliza en una vista o desencadenante, no se puede eliminar y la revocación (REVOKE) está restringida a consecuencia de ello. En otro caso, la revocación se propaga y elimina esas funciones.

## Notas

- Es válido eliminar una función definida por el usuario mientras se está utilizando. También, un cursor puede abrirse en una sentencia que contenga

## DROP

una referencia a una función definida por el usuario y mientras el cursor está abierto la función puede eliminarse sin provocar que fallen las lecturas del cursor.

- Si se está ejecutando un paquete que depende de una función definida por el usuario, no es posible que otro ID de autorización elimine la función hasta que el paquete complete su unidad de trabajo actual. En dicho momento, se elimina la función y el paquete se convierte en no operativo. La siguiente petición de este paquete dará como resultado un error indicando que el paquete debe volverse a enlazar explícitamente.
- La eliminación de un cuerpo de función (es muy diferente de eliminar la función) puede producirse mientras se está ejecutando una aplicación que necesite el cuerpo de la función. Esto puede ocasionar o no que la sentencia falle, según si el gestor de bases de datos tenga que cargar todavía el cuerpo de la función en el almacenamiento para la sentencia.
- En cualquier tabla eliminada que incluyera archivos actualmente enlazados mediante columnas DATALINK, los archivos se desenlazan y se restaurarán o se suprimirán según la definición de la columna DATALINK.
- Si una tabla que contiene una columna DATALINK se elimina (mediante DROP TABLE o DROP TABLESPACE) mientras los DB2 Data Links Managers configurados en la base de datos no se encuentran disponibles, fallará la operación (SQLSTATE 57050).
- Además de las dependencias registradas para cualquier UDF especificada explícitamente, se registran las dependencias siguientes cuando se solicitan transformaciones implícitamente:
  1. Cuando el parámetro de tipo estructurado o el resultado de una función o método solicita una transformación, se registra una dependencia para la función o método respecto a la función de transformación necesaria TO SQL o FROM SQL.
  2. Cuando una sentencia de SQL incluida en un paquete solicita una función de transformación, se registra una dependencia para el paquete respecto a la función de transformación TO SQL o FROM SQL indicada.

Debido a que las condiciones descritas anteriormente son los únicos casos en que se registran dependencias debido a la invocación implícita de transformaciones, las funciones, métodos y paquetes son los únicos objetos que pueden tener una dependencia respecto a funciones de transformación invocadas implícitamente. En cambio, las llamadas explícitas a funciones de transformación (en vistas y desencadenantes, por ejemplo) sí que producen las dependencias habituales de estos otros tipos de objetos respecto a funciones de transformación. Como resultado, una sentencia DROP TRANSFORM puede también fallar debido a estas dependencias de tipo "explícito" que los objetos tienen respecto a las transformaciones que están eliminando (SQLSTATE 42893).

- Debido a que los catálogos de dependencias no distinguen entre el depender de una función en calidad de transformación y el depender de una función por llamada explícita, es recomendable no escribir llamadas explícitas a funciones de transformación. En tal caso, el atributo de transformación de la función no se puede eliminar, o los paquetes se marcan como no operativos, simplemente porque contienen invocaciones explícitas en una expresión SQL.
- Las secuencias creadas por el sistema para las columnas IDENTITY no se pueden eliminar utilizando el mandato de secuencia DROP.
- Cuando se elimina una secuencia, se eliminan también todos los privilegios sobre la secuencia.

## Ejemplos

*Ejemplo 1:* Elimine la tabla TDEPT.

```
DROP TABLE TDEPT
```

*Ejemplo 2:* Elimine la vista VDEPT.

```
DROP VIEW VDEPT
```

*Ejemplo 3:* El ID de autorización HEDGES intenta eliminar un seudónimo.

```
DROP ALIAS A1
```

El seudónimo HEDGES.A1 se elimina de los catálogos.

*Ejemplo 4:* Hedges intenta eliminar un seudónimo, pero especifica T1 como el nombre-seudónimo, cuando T1 es el nombre de una tabla existente (no el nombre de un seudónimo).

```
DROP ALIAS T1
```

Esta sentencia falla (SQLSTATE 42809).

*Ejemplo 5:*

Elimine el grupo de nodos BUSINESS\_OPS. Para eliminarlo, deben eliminarse primero los dos espacios de tablas (ACCOUNTING y PLANS) del grupo de nodos.

```
DROP TABLESPACE ACCOUNTING
DROP TABLESPACE PLANS
DROP NODEGROUP BUSINESS_OPS
```

*Ejemplo 6:* Pellow desea eliminar la función CENTRE, que ha creado en su esquema PELLOW, usando la signatura para identificar la instancia de la función que se ha de eliminar.

```
DROP FUNCTION CENTRE (INT, FLOAT)
```

## DROP

*Ejemplo 7:* McBride desea eliminar la función FOCUS92, que ha creado en el esquema PELLOW, utilizando el nombre específico para identificar la instancia de función que se ha de eliminar.

```
DROP SPECIFIC FUNCTION PELLOW.FOCUS92
```

*Ejemplo 8:* Elimine la función ATOMIC\_WEIGHT del esquema CHEM, donde se sabe que sólo hay una función con dicho nombre.

```
DROP FUNCTION CHEM.ATOMIC_WEIGHT
```

*Ejemplo 9:* Elimine el desencadenante SALARY\_BONUS, que ha provocado que los empleados bajo una condición especificada recibiesen una bonificación en su salario.

```
DROP TRIGGER SALARY_BONUS
```

*Ejemplo 10:* Elimine el tipo de datos diferenciado denominado shoesize, si no se utiliza actualmente.

```
DROP DISTINCT TYPE SHOESIZE
```

*Ejemplo 11:* Elimine el supervisor de sucesos SMITHPAY.

```
DROP EVENT MONITOR SMITHPAY
```

*Ejemplo 12:* Elimine el esquema del Ejemplo 2 bajo CREATE SCHEMA utilizando RESTRICT. Tenga en cuenta que primero debe eliminarse la tabla llamada PART.

```
DROP TABLE PART
DROP SCHEMA INVENTORY RESTRICT
```

*Ejemplo 13:* Macdonald desea eliminar el procedimiento DESTROY, que ha creado en el esquema EIGLER, utilizando el nombre específico para identificar la instancia de procedimiento que se ha de eliminar.

```
DROP SPECIFIC PROCEDURE EIGLER.DESTROY
```

*Ejemplo 14:* Elimine el procedimiento OSMOSIS del esquema BIOLOGY, donde se sabe que sólo hay un procedimiento con dicho nombre.

```
DROP PROCEDURE BIOLOGY.OSMOSIS
```

*Ejemplo 15:* El usuario SHAWN ha utilizado un ID de autorización para acceder a la base de datos federada y otro para acceder a la base de datos en la fuente de datos Oracle llamada ORACLE1. Se ha creado una correlación entre las dos autorizaciones, pero SHAWN ya no necesita acceder a la fuente de datos. Eliminar la correlación.

```
DROP USER MAPPING FOR SHAWN SERVER ORACLE1
```

*Ejemplo 16:* Se ha suprimido un índice de una tabla de fuente de datos al que un apodo hace referencia. Elimine la especificación de índice que se ha creado para dejar que optimizador conozca este índice.

```
DROP INDEX INDEXSPEC
```

*Ejemplo 17:* Eliminación del grupo de transformación MYSTRUCT1 .

```
DROP TRANSFORM MYSTRUCT1 FOR POLYGON
```

*Ejemplo 18:* Eliminación del método BONUS para el tipo de datos EMP en el esquema PERSONNEL.

```
DROP METHOD BONUS (SALARY DECIMAL(10,2)) FOR PERSONNEL.EMP
```

*Ejemplo 19:* Elimine la secuencia ORG\_SEQ, con restricciones.

```
DROP SEQUENCE ORG_SEQ RESTRICT
```

## END DECLARE SECTION

---

### END DECLARE SECTION

La sentencia END DECLARE SECTION marca el final de una sección de declaración de variables de lenguaje principal.

#### Invocación

Esta sentencia sólo puede incluirse en un programa de aplicación. No es una sentencia ejecutable. No debe especificarse en REXX.

#### Autorización

No se necesita.

#### Sintaxis

▶—END DECLARE SECTION—▶

#### Descripción

La sentencia END DECLARE SECTION puede codificarse en el programa de aplicación donde pueda haber declaraciones de acuerdo a las reglas del lenguaje principal. Indica el final de una sección de declaración de variables de lenguaje principal. Una sección de variables de lenguaje principal empieza por una sentencia BEGIN DECLARE SECTION (vea “BEGIN DECLARE SECTION” en la página 615).

Las sentencias BEGIN DECLARE SECTION y END DECLARE SECTION deben especificarse por pares y no pueden anidarse.

Las declaraciones de variables de lenguaje principal pueden especificarse utilizando la sentencia de SQL INCLUDE. De lo contrario, la sección de declaración de variables de lenguaje principal no debe contener ninguna sentencia que no sean declaraciones de variables de lenguaje principal.

Las variables de lenguaje principal referenciadas en sentencias de SQL deben declararse en una sección de declaración de variables en todos los lenguajes principales, excepto REXX.<sup>98</sup> Además, la declaración de cada variable debe preceder a la primera referencia a la variable.

Las variables declaradas fuera de una sección de declaración no deben tener el mismo nombre que las variables declaradas dentro de una sección de declaración.

---

98. Vea “Normas” en la página 615 para conocer cómo se declaran variables de lenguaje principal en REXX cuando se utilizan localizadores de LOB y variables de referencia a archivos.



**Ejemplo**

Vea “BEGIN DECLARE SECTION” en la página 615 para ver ejemplos que utilizan la sentencia END DECLARE SECTION.

## EXECUTE

---

## EXECUTE

La sentencia EXECUTE ejecuta una sentencia de SQL preparada.

### Invocación

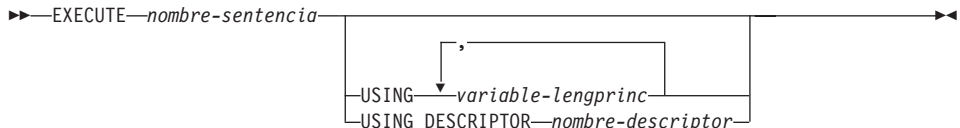
Esta sentencia sólo puede incluirse en un programa de aplicación. Se trata de una sentencia ejecutable que no puede prepararse de forma dinámica.

### Autorización

Para las sentencias donde el control de autorizaciones se realiza durante la ejecución (sentencias DDL, GRANT y REVOKE), los privilegios del ID de autorización de la sentencia deben incluir aquellos necesarios para ejecutar la sentencia de SQL especificada por la sentencia PREPARE. El ID de autorización de la sentencia puede verse afectado por la opción de enlace DYNAMICRULES. Consulte el apartado “Características del SQL dinámico en la ejecución” en la página 81.

Para las sentencias donde el control de autorizaciones se realiza al preparar la sentencia (DML), no es necesaria ninguna autorización para utilizar esta sentencia.

### Sintaxis



### Descripción

#### *nombre-sentencia*

Identifica la sentencia preparada que se ha de ejecutar. El *nombre-sentencia* debe identificar una sentencia que se ha preparado con anterioridad y la sentencia preparada no debe ser una sentencia SELECT.

#### USING

Introduce una lista de variables del lenguaje principal cuyos valores se sustituyen por los marcadores de parámetros (signos de interrogación) en la sentencia preparada. (Para obtener una explicación de marcadores de parámetros, consulte el apartado “PREPARE” en la página 1099.) Si la sentencia preparada incluye los marcadores de parámetros, se ha de utilizar USING.

#### *variable-lengprinc, ...*

Identifica una variable del lenguaje principal que se declara en el programa de acuerdo a las reglas para la declaración de variables del lenguaje principal. El número de variables debe ser igual al número

de marcadores de parámetros de la sentencia preparada. La variable  $n$  corresponde al marcador de parámetro  $n$  de la sentencia preparada.

**DESCRIPTOR** *nombre-descriptor*

Identifica una SQLDA de entrada que debe contener una descripción válida de variables del lenguaje principal.

Antes de procesar la sentencia EXECUTE, el usuario debe establecer los campos siguientes en la SQLDA de entrada:

- SQLN para indicar el número de apariciones de SQLVAR proporcionadas en la SQLDA
- SQLDABC para indicar el número de bytes de almacenamiento asignados para la SQLDA
- SQLD para indicar el número de variables utilizadas en la SQLDA al procesar la sentencia
- Las apariciones de SQLVAR, para indicar los atributos de las variables.

La SQLDA debe tener suficiente almacenamiento para contener todas las ocurrencias de SQLVAR. Por lo tanto, el valor de SQLDABC debe ser mayor o igual que  $16 + \text{SQLN} * (\text{N})$ , donde N es la longitud de una ocurrencia SQLVAR.

Si los datos de entrada LOB necesitan acomodarse, debe haber dos entradas SQLVAR para cada marcador de parámetros.

SQLD debe establecerse en un valor mayor o igual que cero y menor o igual que SQLN. Para obtener más información, vea "Apéndice C. Área de descriptores SQL (SQLDA)" en la página 1267.

## Notas

- Antes de ejecutar la sentencia preparada, cada marcador de parámetros se sustituye efectivamente por el valor de su variable del lenguaje principal correspondiente. Para un marcador de parámetros con tipo, los atributos de la variable de destino son aquellos especificados por la especificación CAST. Para un marcador de parámetros sin tipo, los atributos de la variable de destino se determinan de acuerdo al contexto del marcador de parámetros. Consulte las reglas que afectan a los marcadores de parámetros en el apartado "Normas" en la página 1100.

Supongamos que V indique una variable del lenguaje principal que corresponda al marcador de parámetros P. El valor de P se asigna a la variable de destino para P de acuerdo con las normas de asignación de un valor a una columna. Por lo tanto:

- V debe ser compatible con el destino.

## EXECUTE

- Si V es una serie, su longitud no debe ser mayor que el atributo de longitud del destino.
- Si V es un número, el valor absoluto de su parte correspondiente al entero no debe ser mayor que el valor absoluto máximo de la parte correspondiente a los enteros del destino.
- Si los atributos de V no son idénticos a los atributos del destino, el valor se convierte para ajustarse a los atributos del destino.

Cuando se ejecuta la sentencia preparada, el valor utilizado en lugar de P es el valor de la variable de destino para P. Por ejemplo, si V es CHAR(6) y el destino es CHAR(8), el valor que se utiliza en lugar de P es el valor de V rellenado con dos blancos.

- ***Antememoria de sentencia de SQL dinámica:***

La información necesaria para ejecutar las sentencias de SQL dinámicas y estáticas se coloca en la antememoria del paquete de bases de datos cuando se hace referencia por primera vez a las sentencias de SQL estáticas o cuando se preparan por primera vez la sentencias de SQL dinámicas. Esta información permanece en la antememoria del paquete hasta que se convierte en no válida, el espacio de antememoria es necesario para otra sentencia o se cierra la base de datos.

Cuando se ejecuta o prepara una sentencia de SQL, la información del paquete relevante a la aplicación que emite la petición se carga del catálogo del sistema en la antememoria del paquete. La sección ejecutable real para la sentencia de SQL individual también se coloca en la antememoria: las secciones SQL estáticas se leen del catálogo del sistema y se colocan en la antememoria de paquetes cuando se hace referencia a la sentencia por primera vez; las secciones SQL dinámicas se colocan directamente en la antememoria después de haberse creado. Las secciones SQL dinámicas pueden crearse por una sentencia explícita como, por ejemplo, una sentencia PREPARE o EXECUTE IMMEDIATE. Una vez creadas, la secciones para sentencias de SQL dinámicas pueden volverse a crear por una preparación implícita de la sentencia realizada por el sistema si la sección original se ha suprimido por razones de gestión de espacio o se ha invalidado debido a cambios en el entorno.

Cada sentencia de SQL se pone en antememoria en un nivel de base de datos y se puede compartir entre aplicaciones. Las sentencias de SQL estáticas se comparten entre aplicaciones que utilizan el mismo paquete; las sentencias de SQL dinámicas se comparten entre aplicaciones que utilizan el mismo entorno de compilación y exactamente el mismo texto de sentencia. El texto de cada sentencia de SQL emitida por una aplicación se pone en antememoria localmente dentro de la aplicación para utilizarlo en caso de que sea necesaria una preparación implícita. Cada sentencia PREPARE del programa de aplicación puede poner en antememoria una sentencia. Todas las sentencias EXECUTE IMMEDIATE de un programa de aplicación

comparten el mismo espacio y sólo existe una sentencia en antememoria para todas estas sentencias EXECUTE IMMEDIATE a la vez. Si se emite múltiples veces la misma sentencia PREPARE o cualquier sentencia EXECUTE IMMEDIATE con una sentencia de SQL distinta cada vez, sólo se pondrá en antememoria la última sentencia para volverla a utilizar. La utilización óptima de la antememoria es emitir un número de sentencias PREPARE distintas una vez al inicio de la aplicación y después emitir una sentencia EXECUTE u OPEN, según sea necesario.

Con la antememoria de sentencias de SQL dinámicas, una vez creada una sentencia, puede volverse a utilizar en múltiples unidades de trabajo sin necesidad de preparar la sentencia de nuevo. El sistema volverá a compilar la sentencia tal como sea necesario si se producen cambios en el entorno.

Los sucesos siguientes son ejemplos de cambios en el entorno o en objetos de datos que pueden originar que las sentencias dinámicas en antememoria se preparen implícitamente en la siguiente petición de PREPARE, EXECUTE, EXECUTE IMMEDIATE o OPEN:

- ALTER NICKNAME
- ALTER SERVER
- ALTER TABLE
- ALTER TABLESPACE
- ALTER TYPE
- CREATE FUNCTION
- CREATE FUNCTION MAPPING
- CREATE INDEX
- CREATE TABLE
- CREATE TEMPORARY TABLESPACE
- CREATE TRIGGER
- CREATE TYPE
- DROP (todos los objetos)
- RUNSTATS en cualquier tabla o índice
- cualquier acción que haga que una vista se convierta en no operativa
- UPDATE de estadísticas en cualquier tabla del catálogo del sistema
- SET CURRENT DEGREE
- SET PATH
- SET QUERY OPTIMIZATION
- SET SCHEMA
- SET SERVER OPTION

La lista siguiente resalta el funcionamiento que se puede esperar de las sentencias de SQL dinámicas en antememoria:

## EXECUTE

- *Peticiones de PREPARE*: Las preparaciones posteriores de la misma sentencia no incurrirán en el coste de compilar la sentencia si la sección todavía es válida. Se devolverán las estimaciones de coste y cardinalidad para la sección en antememoria actual. Estos valores pueden diferir de los valores devueltos de las sentencias PREPARE anteriores para la misma sentencia de SQL.

No habrá necesidad de emitir una sentencia PREPARE subsiguiente a una sentencia COMMIT o ROLLBACK.

- *Peticiones de EXECUTE*: Las sentencias EXECUTE pueden incurrir ocasionalmente en el coste de preparar implícitamente la sentencia si se ha convertido en no válida desde la sentencia PREPARE original. Si una sección se prepara implícitamente, utilizará el entorno actual y no el entorno de la sentencia PREPARE original.
- *Peticiones de EXECUTE IMMEDIATE*: Las sentencias EXECUTE IMMEDIATE posteriores para la misma sentencia no incurrirán en el coste de compilar la sentencia si la sección todavía es válida.
- *Peticiones de OPEN*: Las peticiones de OPEN para cursores definidos dinámicamente pueden incurrir ocasionalmente en el coste de preparar implícitamente la sentencia si ya no es válida desde la sentencia PREPARE original. Si una sección se prepara implícitamente, utilizará el entorno actual y no el entorno de la sentencia PREPARE original.
- *Peticiones de FETCH*: No debe esperarse ningún cambio en el funcionamiento.
- *ROLLBACK*: Sólo se invalidarán las sentencias de SQL dinámicas preparadas o implícitamente preparadas durante la unidad de trabajo afectada por la operación de retrotracción.
- *COMMIT*: Las sentencias de SQL dinámicas no se invalidarán pero se liberarán los bloqueos adquiridos. Los cursores que no se hayan definido como cursores WITH HOLD se cerrarán y se liberarán sus bloqueos. Los cursores WITH HOLD abiertos mantendrán sus bloqueos de paquete y de sección para proteger la sección activa durante, y después, del proceso de confirmación.

Si se produce un error durante una preparación implícita, se devolverá un error para la petición que provoca la preparación implícita (SQLSTATE 56098).

### Ejemplos

*Ejemplo 1:* En este ejemplo C, se prepara y ejecuta una sentencia INSERT con marcadores de parámetros. h1 - h4 son variables del lenguaje principal que corresponden al formato de TDEPT.

```
strcpy (s,"INSERT INTO TDEPT VALUES(?,?,?,?)");
EXEC SQL PREPARE DEPT_INSERT FROM :s;
:
:
```

## EXECUTE

(Compruebe la ejecución satisfactoria y ponga valores en :h1, :h2, :h3, :h4)

.

```
EXEC SQL EXECUTE DEPT_INSERT USING :h1, :h2,
:h3, :h4;
```

*Ejemplo 2:* La sentencia EXECUTE utiliza una SQLDA.

```
EXECUTE S3 USING DESCRIPTOR :sqlda3
```

## EXECUTE IMMEDIATE

---

### EXECUTE IMMEDIATE

La sentencia EXECUTE IMMEDIATE:

- Prepara una forma ejecutable de una sentencia de SQL a partir de una forma de sentencia de serie de caracteres.
- Ejecuta la sentencia de SQL.

EXECUTE IMMEDIATE combina las funciones básicas de las sentencias PREPARE y EXECUTE. Puede utilizarse para preparar y ejecutar sentencias de SQL que no contengan variables del lenguaje principal ni marcadores de parámetros.

#### Invocación

Esta sentencia sólo puede incluirse en un programa de aplicación. Se trata de una sentencia ejecutable que no puede prepararse de forma dinámica.

#### Autorización

Las reglas de autorización son las definidas para la sentencia de SQL especificada por EXECUTE IMMEDIATE.

El ID de autorización de la sentencia puede verse afectado por la opción de enlace DYNAMICRULES. Consulte el apartado “Características del SQL dinámico en la ejecución” en la página 81.

#### Sintaxis

►—EXECUTE IMMEDIATE—*variable-lengprinc*—◄

#### Descripción

*variables del lenguaje principal*

Debe especificarse una variable del lenguaje principal y debe identificar una variable del lenguaje principal que se describa en el programa de acuerdo con las reglas para la declaración de variables de serie de caracteres. Debe ser una variable de serie de caracteres menor que el tamaño máximo de la sentencia (65 535). Observe que un CLOB(65535) puede contener una sentencia de tamaño máximo, pero un VARCHAR no puede.

El valor de la variable del lenguaje principal identificada se denomina serie de sentencia.

La serie de sentencia debe ser una de las siguientes sentencias de SQL:

- ALTER
- COMMENT ON
- COMMIT



- CREATE
- DELETE
- DECLARE GLOBAL TEMPORARY TABLE
- DROP
- GRANT
- INSERT
- LOCK TABLE
- REFRESH TABLE
- RELEASE SAVEPOINT
- RENAME TABLE
- RENAME TABLESPACE
- REVOKE
- ROLLBACK
- SAVEPOINT
- SET CURRENT DEGREE
- SET CURRENT EXPLAIN MODE
- SET CURRENT EXPLAIN SNAPSHOT
- SET CURRENT QUERY OPTIMIZATION
- SET CURRENT REFRESH AGE
- SET CURRENT TRANSFORM GROUP
- SET EVENT MONITOR STATE
- SET INTEGRITY
- SET PASSTHRU
- SET PATH
- SET SCHEMA
- SET SERVER OPTION
- UPDATE

La serie de sentencia no debe incluir marcadores de parámetros ni referencias a variables del lenguaje principal, y no debe empezar por EXEC SQL. No debe contener un terminador de sentencia, con la excepción de la sentencia CREATE TRIGGER, que puede contener un punto y coma (;) para separar las sentencias de SQL activadas, o la sentencia CREATE PROCEDURE, para separar las sentencias de SQL en el cuerpo del procedimiento SQL.

Cuando se ejecuta una sentencia EXECUTE IMMEDIATE, la serie de sentencia se analiza y se comprueba si hay errores. Si la sentencia de SQL no es válida, no se ejecuta y la condición de error que impide su ejecución se informa a la

## EXECUTE IMMEDIATE

SQLCA. Si la sentencia de SQL es válida, pero se produce un error durante su ejecución, dicha condición de error se informa a la SQLCA.

### Notas

- El poner en antememoria las sentencias afecta al funcionamiento de la sentencia EXECUTE IMMEDIATE. Consulte "Antememoria de sentencia de SQL dinámica" en la página 1038 para obtener más información.

### Ejemplo

Utilice sentencias de programa C para mover una sentencia de SQL a la variable del lenguaje principal qstring (char[80]) y prepare y ejecute la sentencia de SQL que haya en la variable del lenguaje principal qstring.

```
if (strcmp(accounts,"BIG") == 0)
 strcpy (qstring,"INSERT INTO WORK_TABLE SELECT *
 FROM EMP_ACT WHERE ACTNO < 100");
else
 strcpy (qstring,"INSERT INTO WORK_TABLE SELECT *
 FROM EMP_ACT WHERE ACTNO >= 100");
.
.
EXEC SQL EXECUTE IMMEDIATE :qstring;
```

## EXPLAIN

La sentencia EXPLAIN captura información acerca del plan de acceso elegido para la sentencia explicable suministrada y coloca esta información en las tablas Explain. (Consulte el “Apéndice K. Tablas de Explain y definiciones” en la página 1459 para obtener información sobre las tablas Explain y las definiciones de tabla.)

Una *sentencia explicable* es una sentencia DELETE, INSERT, SELECT, SELECT INTO, UPDATE, VALUES o VALUES INTO SQL.

### Invocación

Esta sentencia puede incluirse en un programa de aplicación o emitirse de forma interactiva. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

La sentencia que se ha de explicar no se ejecuta.

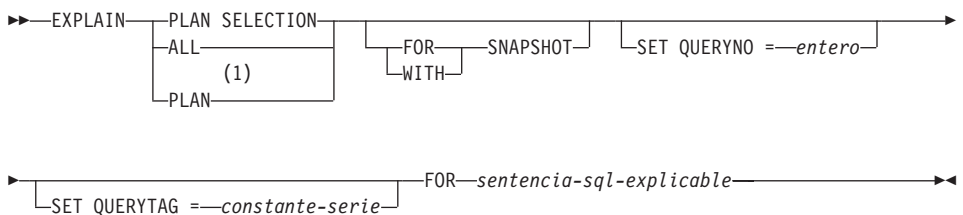
### Autorización

Las reglas de autorización son las definidas para la sentencia de SQL especificada en la sentencia EXPLAIN. Por ejemplo, si se ha utilizado una sentencia DELETE como *sentencia-sql-explicable* (consulte la sintaxis de la sentencia a continuación), las reglas de autorización para una sentencia DELETE se aplican cuando se explica la sentencia DELETE.

Las reglas de autorización para las sentencias EXPLAIN estáticas son las reglas que se aplican a las versiones estáticas de la sentencia pasada como la *sentencia-sql-explicable*. Las sentencias EXPLAIN preparadas dinámicamente utilizan las reglas de autorización proporcionadas para el parámetro *sentencia-sql-explicable*.

El ID de autorización actual debe tener privilegio de inserción para las tablas Explain.

### Sintaxis



### Notas:

- 1 Sólo se da soporte a la opción PLAN para la tolerancia de la sintaxis de

## EXPLAIN

sentencias EXPLAIN existentes de DB2 para MVS. No hay ninguna tabla PLAN. La especificación de PLAN es equivalente a especificar PLAN SELECTION.

### Descripción

#### PLAN SELECTION

Indica que la información de la fase de selección del plan de compilación SQL se ha de insertar en las tablas Explain.

#### ALL

La especificación de ALL es equivalente a especificar PLAN SELECTION.

#### PLAN

La opción PLAN proporciona la tolerancia de la sintaxis para las aplicaciones de bases de datos existentes de otros sistemas. La especificación de PLAN es equivalente a especificar PLAN SELECTION.

#### FOR SNAPSHOT

Esta cláusula indica que sólo se ha de tomar una instantánea de Explain y colocarla en la columna SNAPSHOT de la tabla EXPLAIN\_STATEMENT. No se captura ninguna otra información de Explain que la presente en las tablas EXPLAIN\_INSTANCE y EXPLAIN\_STATEMENT.

La información de la instantánea de Explain está pensada para utilizarla con Visual Explain.

#### WITH SNAPSHOT

Esta cláusula indica que, además de la información de Explain normal, se ha de tomar una instantánea de Explain.

El funcionamiento por omisión de la sentencia EXPLAIN es reunir sólo la información de Explain normal y no la instantánea de Explain.

La información de la instantánea de Explain está pensada para utilizarla con Visual Explain.

por omisión (no se especifican FOR SNAPSHOT ni WITH SNAPSHOT)

Pone la información de Explain en las tablas Explain. No se toma ninguna instantánea para utilizarla con Visual Explain.

#### SET QUERYNO = *entero*

Asocia el *entero*, a través de la columna QUERYNO de la tabla EXPLAIN\_STATEMENT, con la *sentencia-sql-explicable*. El valor entero suministrado debe ser un valor positivo.

Si no se especifica esta cláusula para una sentencia EXPLAIN dinámica, se asigna un valor por omisión de uno (1). Para una sentencia EXPLAIN estática, el valor por omisión asignado es el número de sentencia asignado por el precompilador.

**SET QUERYTAG = constante-serie**

Asocia la *constante-serie*, mediante la columna QUERYTAG de la tabla EXPLAIN\_STATEMENT, con la *sentencia-sql-explicable*. La *constante-serie* puede ser cualquier serie de caracteres de un máximo de 20 bytes de longitud. Si el valor suministrado es inferior a 20 bytes de longitud, el valor se rellena por la derecha con blancos hasta la longitud necesaria.

Si no se especifica esta cláusula para una sentencia EXPLAIN, se utilizan blancos como el valor por omisión.

**FOR sentencia-sql-explicable**

Especifica la sentencia de SQL que se ha de explicar. Esta sentencia puede ser cualquier sentencia DELETE, INSERT, SELECT, SELECT INTO, UPDATE, VALUES o VALUES INTO SQL. Si la sentencia EXPLAIN está incorporada en un programa, la *sentencia-sql-explicable* puede contener referencias a las variables de lenguaje principal (estas variables deben estar definidas en el programa). De manera similar, si se está preparando EXPLAIN dinámicamente, la *sentencia-sql-explicable* puede contener marcadores de parámetros.

La *sentencia-sql-explicable* debe ser una sentencia de SQL válida que podría prepararse y ejecutarse independientemente de la sentencia EXPLAIN. No puede ser un nombre de sentencia ni una variable del lenguaje principal. Las sentencias de SQL que hacen referencia a los cursores definidos a través de CLP no son válidos para utilizarlos con esta sentencia.

Para explicar el SQL dinámico dentro de una aplicación, toda la sentencia EXPLAIN debe estar preparada dinámicamente.

**Notas**

La tabla siguiente muestra la interacción de las palabras clave de instantánea y la información de Explain.

| Palabra clave especificada | ¿Capturar información de Explain? | ¿Tomar una instantánea para Visual Explain? |
|----------------------------|-----------------------------------|---------------------------------------------|
| ninguna                    | Sí                                | No                                          |
| FOR SNAPSHOT               | No                                | Sí                                          |
| WITH SNAPSHOT              | Sí                                | Sí                                          |

Si no se especifica la cláusula FOR SNAPSHOT ni WITH SNAPSHOT, entonces no se toma ninguna instantánea de Explain.

Las tablas Explain deben crearse por el usuario antes de la invocación de EXPLAIN. (Consulte el “Apéndice K. Tablas de Explain y definiciones” en la página 1459 para obtener información sobre las tablas Explain y las

## EXPLAIN

definiciones de tabla.) La información generada por esta sentencia se almacena en estas tablas Explain en el esquema designado cuando se compila la sentencia.

Si no se produce ningún error durante la compilación de la *sentencia-sql-explicable* suministrada, entonces no se almacena información en las tablas Explain.

El plan de acceso generado para la *sentencia-sql-explicable* no se guarda y, por lo tanto, no se puede invocar después. La información de Explain para la *sentencia-sql-explicable* se inserta cuando se compila la sentencia EXPLAIN en sí.

Para una sentencia EXPLAIN de SQL estático, la información se inserta en las tablas Explain durante el enlace y al volver a enlazar explícitamente (vea REBIND en el manual *Consulta de mandatos*). Durante la precompilación, se comentan las sentencias EXPLAIN estáticas en el archivo fuente de la aplicación modificado. En el momento del enlace, las sentencias EXPLAIN del catálogo SYSCAT.STATEMENTS. Cuando se ejecuta el paquete, la sentencia EXPLAIN no se ejecuta. Tenga en cuenta que los números de sección para todas las sentencias de la aplicación serán secuenciales e incluirán las sentencias EXPLAIN. Una alternativa a utilizar una sentencia EXPLAIN estática es utilizar una combinación de las opciones EXPLAIN y EXPLSNAP BIND/PREP. Las sentencias EXPLAIN estáticas se pueden utilizar para hacer que las tablas Explain se llenen para una sentencia de SQL estática específica entre muchas; simplemente ponga un prefijo en la sentencia de destino con la sintaxis de sentencia EXPLAIN adecuada y enlace la aplicación sin utilizar las opciones BIND/PREP de Explain. La sentencia EXPLAIN también se puede utilizar cuando es ventajoso establecer el campo QUERYNO o QUERYTAG en el momento de la invocación de Explain real.

Para las sentencias EXPLAIN de SQL con enlace incremental, las tablas Explain se llenan de datos cuando se somete a compilación la sentencia EXPLAIN. Cuando se ejecuta el paquete, la sentencia EXPLAIN no realiza ningún proceso (aunque se ejecutará correctamente). Cuando las tablas Explain se llenan con datos, el calificador de la tabla Explain y el ID de autorización utilizado durante el llenado de datos serán los pertenientes al propietario del paquete. La sentencia EXPLAIN también se puede utilizar cuando es ventajoso establecer el campo QUERYNO o QUERYTAG en el momento de la invocación de Explain real.

Para sentencias EXPLAIN dinámicas, las tablas Explain se llenan en el momento que la sentencia EXPLAIN se somete a la compilación. Una sentencia Explain puede prepararse con la sentencia PREPARE, pero su ejecución no realizará ningún proceso (aunque la sentencia se ejecutará correctamente). Una alternativa a la emisión de sentencias EXPLAIN

dinámicas es utilizar una combinación de los registros especiales CURRENT EXPLAIN MODE y CURRENT EXPLAIN SNAPSHOT para explicar las sentencias de SQL dinámicas. La sentencia EXPLAIN debe utilizarse cuando es ventajoso establecer el campo QUERYNO o QUERYTAG en el momento de la invocación de Explain real.

## Ejemplos

*Ejemplo 1:* Explique una sentencia SELECT simple y póngale el distintivo QUERYNO = 13.

```
EXPLAIN PLAN SET QUERYNO = 13 FOR SELECT C1 FROM T1;
```

Esta sentencia es satisfactoria.

*Ejemplo 2:*

Explique una sentencia SELECT simple y póngale un distintivo QUERYTAG = 'TEST13'.

```
EXPLAIN PLAN SELECTION SET QUERYTAG = 'TEST13'
FOR SELECT C1 FROM T1;
```

Esta sentencia es satisfactoria.

*Ejemplo 3:* Explique una sentencia SELECT simple y póngale los distintivos QUERYNO = 13 y QUERYTAG = 'TEST13'.

```
EXPLAIN PLAN SELECTION SET QUERYNO = 13 SET QUERYTAG = 'TEST13'
FOR SELECT C1 FROM T1;
```

Esta sentencia es satisfactoria.

*Ejemplo 4:* Intente obtener información de Explain cuando no existen las tablas de Explain.

```
EXPLAIN ALL FOR SELECT C1 FROM T1;
```

Esta sentencia fallaría ya que no se han definido las tablas de Explain (SQLSTATE 42704).

## FETCH

---

## FETCH

La sentencia FETCH posiciona el cursor en la siguiente fila de su tabla resultante y asigna los valores de dicha fila a las variables del lenguaje principal.

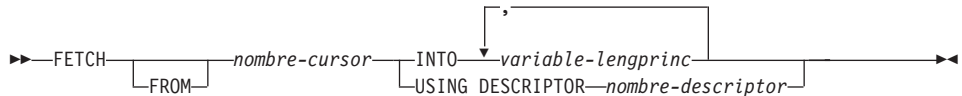
### Invocación

Aunque un recurso SQL interactivo pueda proporcionar una interfaz que dé la apariencia de ejecución interactiva, esta sentencia sólo puede incluirse dentro de un programa de aplicación. Se trata de una sentencia ejecutable que no puede prepararse de forma dinámica.

### Autorización

Vea “DECLARE CURSOR” en la página 976 para ver una explicación de la autorización necesaria para utilizar un cursor.

### Sintaxis



### Descripción

#### *nombre-cursor*

Identifica el cursor que se va a utilizar en una operación de lectura. El *nombre-cursor* debe identificar un cursor declarado tal como se explica en el apartado “DECLARE CURSOR” en la página 976. La sentencia DECLARE CURSOR debe preceder a la sentencia FETCH en el programa fuente. Cuando se ejecuta la sentencia FETCH, el cursor debe estar en el estado abierto.

Si el cursor está situado actualmente en la última fila o después de ella en la tabla resultante:

- SQLCODE se establece en +100 y SQLSTATE se establece en '02000'.
- El cursor se sitúa después de la última fila.
- Los valores no se asignan a las variables del lenguaje principal.

Si el cursor está situado actualmente antes de una fila, se volverá a situar en dicha fila y se asignarán los valores a las variables del lenguaje principal tal como especifican INTO o USING.

Si el cursor está situado actualmente en una fila que no es la última, se situará en la siguiente fila y se asignarán los valores de dicha fila a las variables del lenguaje principal tal como se especifican INTO o USING.



**INTO** *variable-lengprinc, ...*

Identifica una o varias variables del lenguaje principal que deben describirse de acuerdo a las reglas para la declaración de las variables del lenguaje principal. El primer valor de la fila del resultado se asigna a la primera variable del lenguaje principal de la lista, el segundo valor a la segunda variable del lenguaje principal, etcétera. Para los valores LOB de la lista de selección, el destino puede ser una variable del lenguaje principal normal (si es lo suficientemente grande), una variable localizadora o una variable de referencia a archivos.

**USING DESCRIPTOR** *nombre-descriptor*

Identifica una SQLDA que debe contener una descripción válida de cero o más variables del lenguaje principal.

Antes de procesar la sentencia FETCH, el usuario debe establecer los campos siguientes en la SQLDA:

- SQLN para indicar el número de apariciones de SQLVAR proporcionadas en la SQLDA.
- SQLDABC para indicar el número de bytes de almacenamiento asignado para la SQLDA.
- SQLD para indicar el número de variables utilizadas en SQLDA al procesar la sentencia.
- Las apariciones SQLVAR, para indicar los atributos de las variables.

La SQLDA debe tener suficiente almacenamiento para contener todas las ocurrencias de SQLVAR. Por lo tanto, el valor de SQLDABC debe ser mayor o igual que  $16 + \text{SQLN} * (\text{N})$ , donde N es la longitud de una ocurrencia SQLVAR.

Si las columnas resultantes de LOB o de tipo estructurado necesitan acomodarse, debe haber dos entradas SQLVAR para cada elemento de la lista de selección (o columna de la tabla resultante). Vea “Efecto de DESCRIBE en la SQLDA” en la página 1274, que trata sobre las columnas SQLDOUBLED, LOB y de tipo estructurado.

SQLD debe establecerse en un valor mayor o igual que cero y menor o igual que SQLN. Para obtener más información, vea “Apéndice C. Área de descriptores SQL (SQLDA)” en la página 1267.

La variable *n* identificada en la cláusula INTO o descrita en la SQLDA corresponde a la columna *n* de la tabla resultante del cursor. El tipo de datos de cada variable debe ser compatible con su columna correspondiente.

Cada asignación a una variable se realiza de acuerdo a las reglas descritas en el Capítulo 3. Si el número de variables es menor que el número de valores de la fila, el campo SQLWARN3 de la SQLDA se establece en 'W'. Tenga en

## FETCH

cuenta que no hay ningún aviso si hay más variables que el número de columnas del resultado. Si se produce un error de asignación, no se asigna el valor a la variable y no se asignan más valores a las variables. Cualquier valor que ya se haya asignado a las variables continúa asignado.

### Notas

- Un cursor abierto tiene tres posiciones posibles:
  - Antes de una fila
  - En una fila
  - Después de la última fila.
- Si un cursor está en una fila, dicha fila se llama la fila actual del cursor. Un cursor al que se haga referencia en una sentencia UPDATE o DELETE debe estar situado en una fila. Un cursor sólo puede estar en una fila como resultado de una sentencia FETCH.
- Cuando se recuperan datos con localizadores de LOB en situaciones en que no es necesario conservar el localizador entre una sentencia FETCH y otra, es aconsejable emitir una sentencia FREE LOCATOR antes de emitir la siguiente sentencia FETCH, ya que los recursos del localizador son limitados.
- Es posible que se produzca un error que haga que el estado del cursor sea imprevisible.
- Es posible que FETCH no devuelva un aviso. También es posible que el aviso devuelto corresponda a una fila recuperada anteriormente. Esto se produce como consecuencia de optimizaciones que hacen uso de tablas temporales del sistema o de operadores de lectura inversa (vea el manual *Administration Guide*).
- La puesta de sentencias en antememoria afecta al funcionamiento de la sentencia EXECUTE IMMEDIATE. Vea el apartado “Notas” en la página 1037 para obtener más información.
- CLI de DB2 da soporte a posibilidades de lectura adicionales. Por ejemplo cuando la tabla resultante de un cursor es de sólo lectura, se puede utilizar la función SQLFetchScroll() para situar el cursor en cualquier punto dentro de dicha tabla resultante.

### Ejemplos

*Ejemplo 1:* En este ejemplo C, la sentencia FETCH coloca los resultados de la sentencia SELECT en las variables de programa dnum, dname y mnum. Cuando ya no quedan más filas para leer, se devuelve la condición de no encontrado.

```
EXEC SQL DECLARE C1 CURSOR FOR
 SELECT DEPTNO, DEPTNAME, MGRNO FROM TDEPT
 WHERE ADMRDEPT = 'A00';
```

```
EXEC SQL OPEN C1;
```

```
 mientras (SQLCODE==0) {
```

```
EXEC SQL FETCH C1 INTO :dnum, :dname, :mnum;
}

EXEC SQL CLOSE C1;
```

*Ejemplo 2:* La sentencia **FETCH** utiliza una **SQLDA**.

```
FETCH CURS USING DESCRIPTOR :sqlda3
```

## FLUSH EVENT MONITOR

---

### FLUSH EVENT MONITOR

La sentencia FLUSH EVENT MONITOR graba los valores actuales del supervisor de bases de datos para todos los tipos de supervisores activos asociados con el supervisor de sucesos *nombre-supervisor-sucesos* en el destino de E/S del supervisor de sucesos. Por lo tanto, en cualquier momento está disponible un registro parcial del suceso para los supervisores de sucesos que tienen una frecuencia baja de generación de registros (por ejemplo, supervisor de sucesos de bases de datos). Estos registros se indican en el registro cronológico del supervisor de sucesos mediante un identificador de *registro parcial*.

Cuando se desecha un supervisor de sucesos, sus almacenamientos intermedios internos activos se graban en el objeto de salida del supervisor de sucesos.

#### Invocación

Esta sentencia puede incluirse en un programa de aplicación o emitirse de forma interactiva. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

#### Autorización

Los privilegios del ID de autorización deben incluir la autorización SYSADM o DBADM (SQLSTATE 42502).

#### Sintaxis

►► FLUSH—EVENT—MONITOR—*nombre-supervisor-sucesos* BUFFER ►►

#### Descripción

*nombre-supervisor-sucesos*

Nombre del supervisor de sucesos. Este nombre sólo se compone de una parte. Es un identificador de SQL.

#### BUFFER

Indica que se han de grabar los almacenamientos intermedios del supervisor de sucesos. Si se especifica BUFFER, no se generan los registros parciales. Sólo se graban los datos que ya están presentes en los almacenamientos intermedios del supervisor de sucesos.

#### Notas

- Cuando se desecha el supervisor de sucesos no se restauran los valores del supervisor de sucesos. Esto significa que el registro del supervisor de sucesos que se habría generado si no se hubiese desechado, se seguirá generando cuando se active el suceso del supervisor normal.

## FREE LOCATOR

La sentencia FREE LOCATOR elimina la asociación entre una variable localizadora y su valor.

### Invocación

Esta sentencia sólo puede incluirse en un programa de aplicación. Se trata de una sentencia ejecutable que no puede prepararse de forma dinámica.

### Autorización

No se necesita.

### Sintaxis



### Descripción

**LOCATOR** *nombre-variable*, ...

Identifica una o varias variables localizadoras que deben declararse de acuerdo con las reglas para la declaración de variables localizadoras.

La variable-localizadora debe tener actualmente un localizador asignado a ella. Es decir, debe haberse asignado un localizador durante esta unidad de trabajo (por una sentencia FETCH o una sentencia SELECT INTO) y no debe haberse liberado posteriormente (por una sentencia FREE LOCATOR); de lo contrario, se genera un error (SQLSTATE 0F001).

Si se especifica más de un localizador, se liberan todos los localizadores que se pueden liberar, sin tener en cuenta los errores detectados en otros localizadores de la lista.

### Ejemplo

En un programa COBOL, libere las variables localizadoras de BLOB, TKN-VIDEO y TKN-BUF, y la variable localizadora de CLOB, LIFE-STORY-LOCATOR.

```
EXEC SQL
FREE LOCATOR :TKN-VIDEO, :TKN-BUF, :LIFE-STORY-LOCATOR
END-EXEC.
```

## GRANT (autorizaciones de base de datos)

### GRANT (autorizaciones de base de datos)

Este formato de la sentencia GRANT otorga las autorizaciones que se aplican a toda la base de datos (en lugar de privilegios que se aplican a objetos específicos de la base de datos).

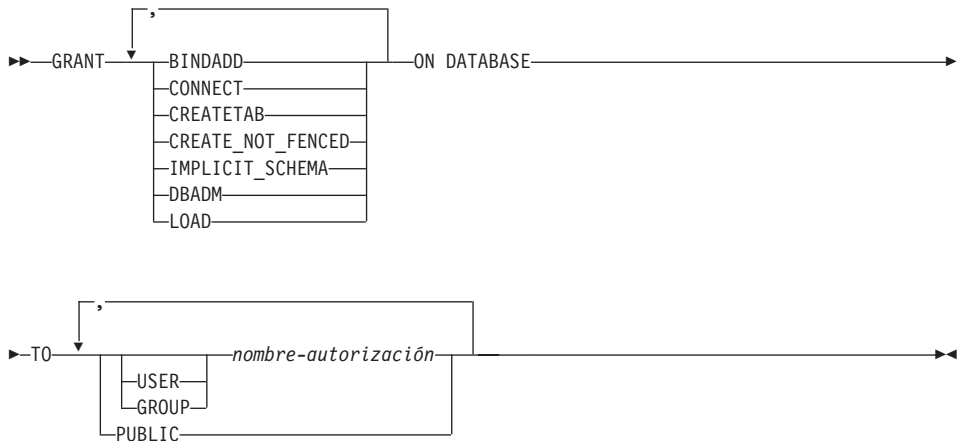
#### Invocación

Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

#### Autorización

Para otorgar la autorización DBADM, se necesita la autorización SYSADM. Para otorgar otras autorizaciones, son necesarias la autorización DBADM o SYSADM.

#### Sintaxis



#### Descripción

##### BINDADD

Otorga la autorización para crear paquetes. El creador de un paquete tiene automáticamente el privilegio CONTROL en dicho paquete y conserva este privilegio incluso si se revoca posteriormente la autorización BINDADD.

##### CONNECT

Otorga la autorización para acceder a la base de datos.

##### CREATETAB

Otorga la autorización para crear tablas base. El creador de una tabla base

## GRANT (autorizaciones de base de datos)

tiene automáticamente el privilegio CONTROL en dicha tabla. El creador conserva este privilegio incluso si posteriormente se revoca la autorización CREATETAB.

No se necesita ninguna autorización explícita para la creación de vistas. Una vista se puede crear en cualquier momento si el ID de autorización de la sentencia utilizada para crear la vista tiene el privilegio CONTROL o SELECT en cada tabla base de la vista.

### CREATE\_NOT\_FENCED

Otorga la autorización para registrar las funciones que se ejecutan en el proceso del gestor de bases de datos. Debe tenerse cuidado de que las funciones registradas de esta manera no tengan efectos negativos (consulte la cláusula FENCED o NOT FENCED en la página 708 para obtener más información.)

Cuando se ha registrado una función como no limitada, continúa ejecutándose de esta manera incluso si se revoca posteriormente CREATE\_NOT\_FENCED.

### IMPLICIT\_SCHEMA

Otorga la autorización para crear implícitamente un esquema.

### DBADM

Otorga la autorización de administrador de la base de datos. El administrador de una base de datos tiene todos los privilegios en todos los objetos de la base de datos y puede otorgar estos privilegios a otros.

BINDADD, CONNECT, CREATETAB, CREATE\_NOT\_FENCED y IMPLICIT\_SCHEMA se otorgan automáticamente a un *nombre-autorización* al que se le otorga autorización DBADM.

### LOAD

Otorga autorización para cargar en la base de datos. Esta autorización proporciona al usuario el derecho a utilizar el programa de utilidad LOAD para la base de datos. Por omisión, SYSADM y DBADM también tienen esta autorización. Sin embargo, si un usuario sólo tiene autorización LOAD (no SYSADM ni DBADM), es necesario que el usuario tenga también privilegios para tablas. Además del privilegio LOAD, el usuario debe tener:

- Privilegio INSERT sobre la tabla para realizar una carga en la modalidad INSERT, TERMINATE (para finalizar un LOAD INSERT anterior) o RESTART (para reiniciar un LOAD INSERT anterior)
- Privilegio INSERT y DELETE sobre la tabla para realizar una carga en la modalidad REPLACE, TERMINATE (para finalizar un LOAD REPLACE anterior) o RESTART (para reiniciar un LOAD REPLACE anterior)
- Privilegio INSERT para la tabla de excepciones, si esa tabla se utiliza como parte de LOAD

## GRANT (autorizaciones de base de datos)

### TO

Especifica a quién se otorgan las autorizaciones.

### USER

Especifica que el *nombre-autorización* identifica a un usuario.

### GROUP

Especifica que el *nombre-autorización* identifica un nombre de grupo.

*nombre-autorización*,...

Lista los ID de autorización de uno o varios usuarios o grupos.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

### PUBLIC

Otorga las autorizaciones a todos los usuarios. DBADM no se puede otorgar a PUBLIC.

## Normas

- Si no se especifica USER ni GROUP, entonces
  - Si se define el nombre-autorización en el sistema operativo sólo como GROUP, entonces se supone GROUP.
  - Si se define el nombre-autorización en el sistema operativo sólo como USER o si no está definido, se supone USER.
  - Si el nombre-autorización está definido en el sistema operativo como ambos, o se utiliza la autenticación DCE, se genera un error (SQLSTATE 56092).

## Ejemplos

*Ejemplo 1:* Otorgue a los usuarios WINKEN, BLINKEN y NOD la autorización para conectarse a la base de datos.

```
GRANT CONNECT ON DATABASE TO USER WINKEN, USER BLINKEN, USER NOD
```

*Ejemplo 2:* Otorgue (GRANT) la autorización BINDADD en la base de datos a un grupo llamado D024. Hay un grupo y un usuario llamados D024 en el sistema.

```
GRANT BINDADD ON DATABASE TO GROUP D024
```

Observe que debe especificarse la palabra clave GROUP; de lo contrario, se producirá un error ya que existen tanto un usuario como un grupo llamados D024. Cualquier miembro del grupo D024 podrá enlazar paquetes en la base de datos, pero el usuario D024 no podrá (a menos que también sea miembro del grupo D024, se le haya otorgado la autorización BINDADD previamente o se haya otorgado la autorización BINDADD a otro grupo del que D024 sea miembro).



## GRANT (privilegios de índice)

Esta forma de la sentencia GRANT otorga el privilegio CONTROL en índices.

### Invocación

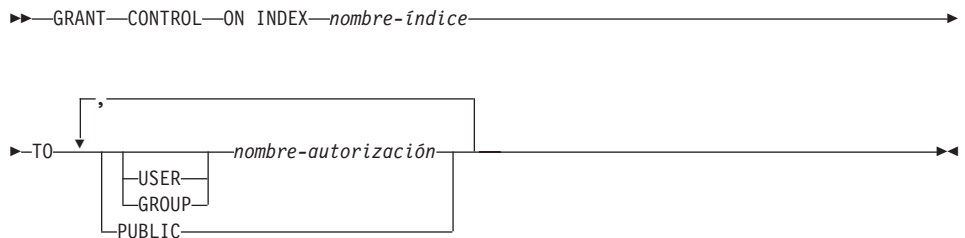
Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

### Autorización

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización DBADM
- Autorización SYSADM.

### Sintaxis



### Descripción

#### CONTROL

Otorga el privilegio para eliminar el índice. Esta es la autorización CONTROL para los índices, que se otorga automáticamente a los creadores de índices.

#### ON INDEX *nombre-índice*

Identifica el índice para el cual se ha de otorgar el privilegio CONTROL.

#### TO

Especifica a quién se otorgan los privilegios.

#### USER

Especifica que el *nombre-autorización* identifica a un usuario.

#### GROUP

Especifica que el *nombre-autorización* identifica un nombre de grupo.

*nombre-autorización*,...

Lista los ID de autorización de uno o varios usuarios o grupos.

## GRANT (privilegios de índice)

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

### **PUBLIC**

Otorga los privilegios a todos los usuarios.

### **Normas**

- Si no se especifica USER ni GROUP, entonces
  - Si se define el nombre-autorización en el sistema operativo sólo como GROUP, entonces se supone GROUP.
  - Si se define el nombre-autorización en el sistema operativo sólo como USER o si no está definido, se supone USER.
  - Si el nombre-autorización está definido en el sistema operativo como ambos, o se utiliza la autenticación DCE, se genera un error (SQLSTATE 56092).

### **Ejemplo**

```
GRANT CONTROL ON INDEX DEPTIDX TO USER USER4
```

---

## GRANT (privilegios de paquete)

Esta forma de la sentencia GRANT otorga los privilegios en un paquete.

### Invocación

Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Es una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

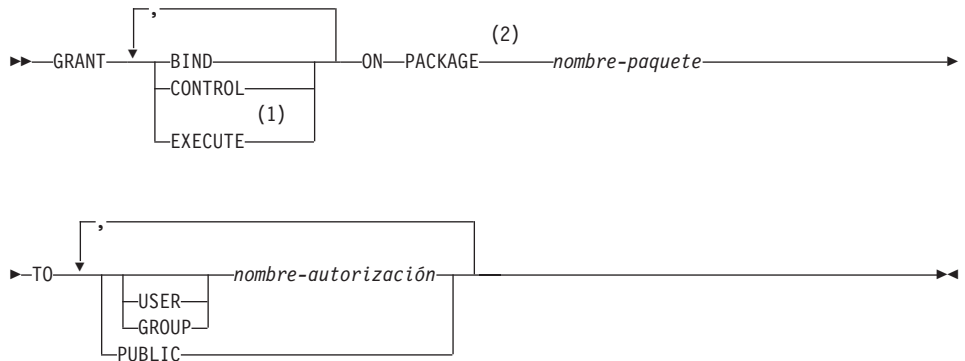
### Autorización

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Privilegio CONTROL para el paquete referenciado
- Autorización SYSADM o DBADM.

Para otorgar el privilegio CONTROL, es necesaria la autorización SYSADM o DBADM.

### Sintaxis



#### Notas:

- 1 RUN se puede utilizar como sinónimo de EXECUTE.
- 2 PROGRAM puede utilizarse como sinónimo de PACKAGE.

### Descripción

#### BIND

Otorga el privilegio para enlazar un paquete. Realmente el privilegio BIND es un privilegio de volver a enlazar, porque el paquete ya debe haberse enlazado (por alguna persona con autorización BINDADD) para poder existir.

## GRANT (privilegios de paquete)

Además del privilegio BIND, el usuario debe poseer los privilegios necesarios en cada tabla a la que hagan referencia las sentencias DML estáticas contenidas en el programa. Esto es necesario porque la autorización en las sentencias DML estáticas se comprueba en el momento del enlace.

### CONTROL

Otorga el privilegio para volver a enlazar, eliminar o ejecutar el paquete y extender los privilegios del paquete a otros usuarios. El privilegio CONTROL para paquetes se otorga automáticamente a los creadores de los paquetes. El propietario de un paquete es el enlazador del paquete o el ID especificado con la opción OWNER durante el enlace/precompilación.

BIND y EXECUTE se otorgan automáticamente a un *nombre-autorización* al que se le otorga el privilegio CONTROL.

### EXECUTE

Otorga el privilegio para ejecutar el paquete.

### ON PACKAGE *nombre-paquete*

Especifica el nombre del paquete en el que se han de otorgar los privilegios.

### TO

Especifica a quién se otorgan los privilegios.

### USER

Especifica que el *nombre-autorización* identifica a un usuario.

### GROUP

Especifica que el *nombre-autorización* identifica un nombre de grupo.  
*nombre-autorización,...*

Lista los ID de autorización de uno o varios usuarios o grupos.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

### PUBLIC

Otorga los privilegios a todos los usuarios.

## Normas

- Si no se especifica USER ni GROUP, entonces
  - Si se define el *nombre-autorización* en el sistema operativo sólo como GROUP, entonces se supone GROUP.
  - Si se define el *nombre-autorización* en el sistema operativo sólo como USER o si no está definido, se supone USER.
  - Si se define el *nombre-autorización* en el sistema operativo como ambos, o se utiliza la autenticación DCE, se genera un error (SQLSTATE 56092).

### Ejemplos

*Ejemplo 1:* Otorgue el privilegio EXECUTE en PACKAGE CORPDATA.PKGA a PUBLIC.

```
GRANT EXECUTE
ON PACKAGE CORPDATA.PKGA
TO PUBLIC
```

*Ejemplo 2:* Otorgue (GRANT) el privilegio EXECUTE en el paquete CORPDATA.PKGA a un usuario denominado EMPLOYEE. No hay ningún grupo ni usuario llamado EMPLOYEE.

```
GRANT EXECUTE ON PACKAGE
CORPDATA.PKGA TO EMPLOYEE
```

o

```
GRANT EXECUTE ON PACKAGE
CORPDATA.PKGA TO USER EMPLOYEE
```

## GRANT (privilegios de esquema)

---

### GRANT (privilegios de esquema)

Esta forma de la sentencia GRANT otorga privilegios en un esquema.

#### Invocación

Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

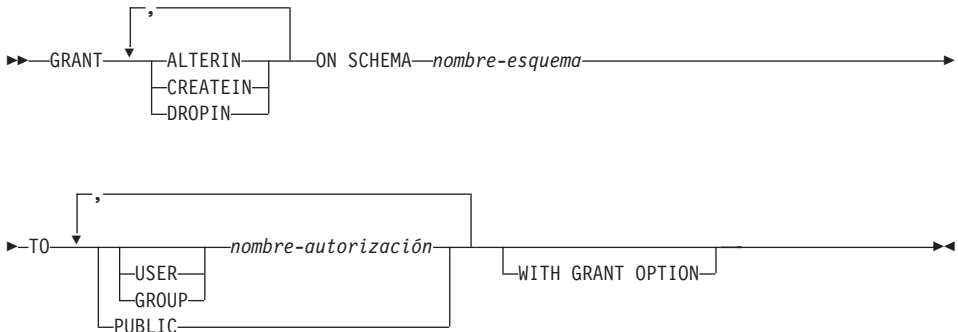
#### Autorización

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- WITH GRANT OPTION para cada privilegio identificado de *nombre-esquema*
- Autorización SYSADM o DBADM

Ningún usuario puede otorgar privilegios en los nombres de esquema SYSIBM, SYSCAT, SYSFUN y SYSSTAT (SQLSTATE 42501).

#### Sintaxis



#### Descripción

##### ALTERIN

Otorga el privilegio para modificar o comentar todos los objetos del esquema. El propietario de un esquema creado explícitamente recibe automáticamente el privilegio ALTERIN.

##### CREATEIN

Otorga el privilegio para crear objetos en el esquema. Siguen necesitándose las demás autorizaciones o privilegios necesarios para crear el objeto (como CREATETAB). El propietario de un esquema creado

## GRANT (privilegios de esquema)

explícitamente recibe automáticamente el privilegio CREATEIN. En un esquema creado implícitamente se otorga automáticamente el privilegio CREATEIN a PUBLIC.

### DROPIN

Otorga el privilegio para eliminar todos los objetos del esquema. El propietario de un esquema creado explícitamente recibe automáticamente el privilegio DROPIN.

### ON SCHEMA *nombre-esquema*

Identifica el esquema en el que se han de otorgar los privilegios.

### TO

Especifica a quién se otorgan los privilegios.

### USER

Especifica que el *nombre-autorización* identifica a un usuario.

### GROUP

Especifica que el *nombre-autorización* identifica un nombre de grupo.

*nombre-autorización,...*

Lista los ID de autorización de uno o varios usuarios o grupos.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

### PUBLIC

Otorga los privilegios a todos los usuarios.

### WITH GRANT OPTION

Permite que los *nombres-autorización* especificados otorguen (GRANT) los privilegios a otros.

Si se omite WITH GRANT OPTION, los *nombres-autorización* sólo pueden otorgar los privilegios a otros si:

- tienen la autorización DBADM o
- han recibido la posibilidad de otorgar privilegios por otra fuente.

## Normas

- Si no se especifica USER ni GROUP, entonces
  - Si se define el nombre-autorización en el sistema operativo sólo como GROUP, entonces se supone GROUP.
  - Si se define el nombre-autorización en el sistema operativo sólo como USER o si no está definido, se supone USER.
  - Si el nombre-autorización está definido en el sistema operativo como ambos, o se utiliza la autenticación DCE, se genera un error (SQLSTATE 56092).
- En general, la sentencia GRANT procesará la operación de otorgar privilegios que el ID de autorización de la sentencia tenga permitido

## GRANT (privilegios de esquema)

otorgar, devolviendo un aviso (SQLSTATE 01007) si no se han otorgado uno o varios privilegios. Si no se ha otorgado ningún privilegio, se devuelve un error (SQLSTATE 42501).<sup>99</sup>

### Ejemplos

*Ejemplo 1:* Otorgue al usuario JSINGLETON la posibilidad de crear objetos en el esquema CORPDATA.

```
GRANT CREATEIN ON SCHEMA CORPDATA TO JSINGLETON
```

*Ejemplo 2:* Otorgue al usuario IHAKES la posibilidad de crear y eliminar objetos en el esquema CORPDATA.

```
GRANT CREATEIN, DROPIN ON SCHEMA CORPDATA TO IHAKES
```

---

<sup>99</sup>. Si el paquete utilizado para procesar la sentencia se ha precompilado con LANGLEVEL establecido en SQL92E para MIA, se devuelve un aviso (SQLSTATE 01007) a menos que el autorizador no tenga ningún privilegio sobre el objeto del otorgamiento.



## GRANT (privilegios de secuencia)

Esta forma de la sentencia GRANT otorga privilegios en una secuencia definida por el usuario.

### Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse dinámicamente. Sin embargo, si se aplica la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

### Autorización

El ID de autorización de la sentencia debe tener al menos uno de los siguientes privilegios:

- Propietario de la secuencia
- Autorización SYSADM o DBADM

### Sintaxis

```
▶▶—GRANT—USAGE—ON SEQUENCE—nombre-secuencia—TO PUBLIC—▶▶
```

### Descripción

#### USAGE

Otorga el privilegio USAGE para una secuencia. El privilegio USAGE en una secuencia es necesario cuando se invoca la expresión NEXTVAL o PREVVVAL con un nombre de secuencia específico.

#### ON SEQUENCE *nombre-secuencia*

Identifica la secuencia en la que se debe otorgar el privilegio USAGE. El *nombre-secuencia*, incluido el calificador de esquema implícito o explícito, debe identificar de forma exclusiva una secuencia existente en el servidor actual. Si no existe ninguna secuencia con este nombre en el esquema especificado, se producirá un error (SQLSTATE 42704).

#### TO PUBLIC

Otorga el privilegio USAGE a todos los usuarios.

### Ejemplos

*Ejemplo 1:* Otorgue a cualquier usuario el privilegio en una secuencia llamada ORG\_SEQ

```
GRANT USAGE ON SEQUENCE ORG_SEQ TO PUBLIC
```

## GRANT (Privilegios de servidor)

---

### GRANT (Privilegios de servidor)

Este formato de la sentencia GRANT otorga el privilegio de acceder y utilizar una fuente de datos especificada en la modalidad de paso a través.

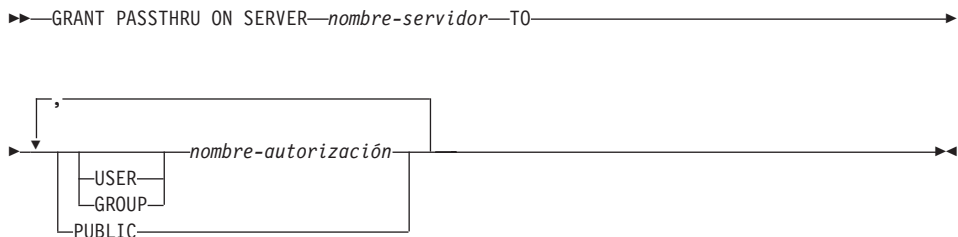
#### Invocación

Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que se puede preparar dinámicamente. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

#### Autorización

El ID de autorización de la sentencia debe tener la autorización SYSADM o DBADM.

#### Sintaxis



#### Descripción

*nombre-servidor*

Designa la fuente de datos para la cual se está otorgando el privilegio que debe utilizarse en la modalidad de paso a través. *nombre-servidor* debe identificar una fuente de datos que esté descrita en el catálogo.

**TO**

Especifica a quién se otorga el privilegio.

**USER**

Especifica que el *nombre-autorización* identifica a un usuario.

**GROUP**

Especifica que el *nombre-autorización* identifica un nombre de grupo.

*nombre-autorización,...*

Lista los ID de autorización de uno o varios usuarios o grupos.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

### PUBLIC

Otorga a todos los usuarios el privilegio de realizar un paso a través para *nombre-servidor*.

### Ejemplos

*Ejemplo 1:* Otorgue a R. Smith y a J. Jones el privilegio de paso a través para la fuente de datos SERVALL. Sus ID de autorización son RSMITH y JJONES.

```
GRANT PASSTHRU ON SERVER SERVALL
TO USER RSMITH,
USER JJONES
```

*Ejemplo 2:* Otorgue el privilegio de realizar un paso a través para la fuente de datos EASTWING a un grupo cuyo ID de autorización es D024. Existe un usuario cuyo ID de autorización también es D024.

```
GRANT PASSTHRU ON SERVER EASTWING TO GROUP D024
```

Debe especificarse la palabra clave GROUP; de lo contrario se producirá un error porque D024 es el ID de un usuario y el ID del grupo especificado (SQLSTATE 56092). Cualquier miembro del grupo D024 tendrá permitido realizar un paso a través para EASTWING. Por lo tanto, si el usuario D024 pertenece al grupo, este usuario podrá realizar un paso a través para EASTWING.

## GRANT (privilegios de apodo, vista o tabla)

### GRANT (privilegios de apodo, vista o tabla)

Esta forma de la sentencia GRANT otorga privilegios en una tabla, vista o apodo.

#### Invocación

Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

#### Autorización

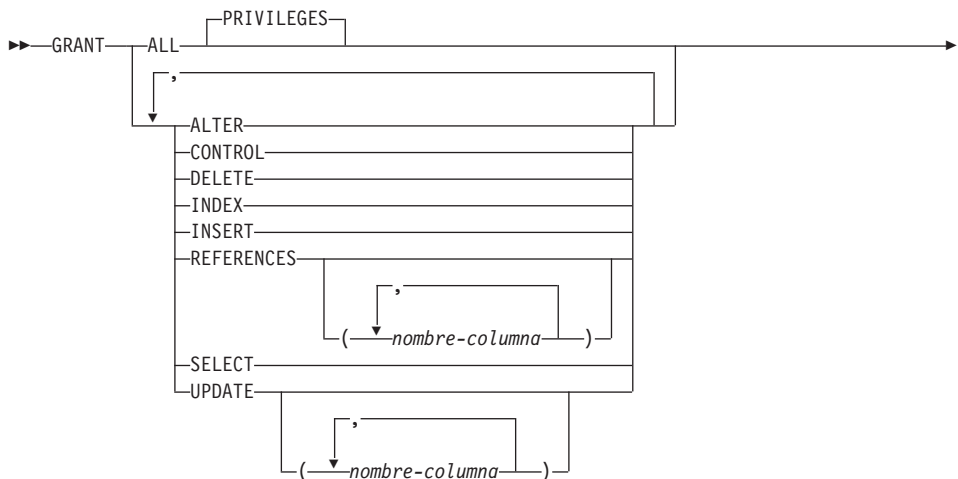
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Privilegio CONTROL para la tabla, vista o apodo referenciado.
- WITH GRANT OPTION para cada privilegio identificado. Si se especifica ALL, el ID de autorización debe tener algún privilegio otorgable en la tabla, vista o apodo identificado.
- Autorización SYSADM o DBADM.

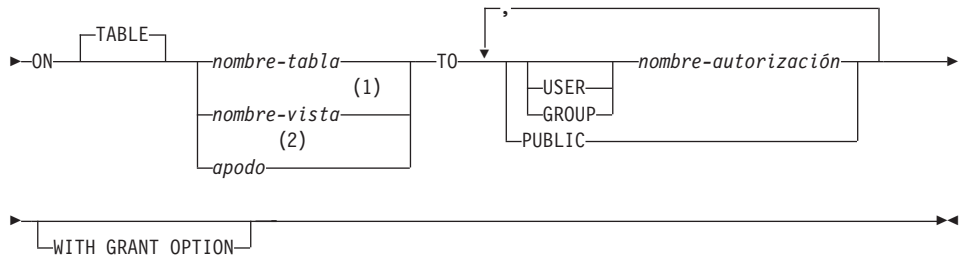
Para otorgar el privilegio CONTROL, es necesaria la autorización SYSADM o DBADM.

Para otorgar privilegios sobre tablas y vistas de catálogo son necesarias las autorizaciones SYSADM o DBADM.

#### Sintaxis



## GRANT (privilegios de apodo, vista o tabla)



### Notas:

- 1 Los privilegios ALTER, INDEX y REFERENCES no son aplicables a las vistas.
- 2 Los privilegios DELETE, INSERT, SELECT y UPDATE no se aplican a los apodos.

### Descripción

#### ALL o ALL PRIVILEGES

Otorga todos los privilegios adecuados, excepto CONTROL, en la tabla base, vista o apodo llamado en la cláusula ON.

Si el ID de autorización de la sentencia tiene el privilegio CONTROL en la tabla, vista o apodo, o la autorización DBADM o SYSADM, entonces se otorgan todos los privilegios aplicables al objeto (excepto CONTROL). De lo contrario, los privilegios otorgados son todos los privilegios otorgables que el ID de autorización de la sentencia tenga en la tabla, vista o apodo identificado.

Si no se especifica ALL, debe especificarse una o varias palabras clave en la lista de privilegios.

#### ALTER

Otorga el privilegio para:

- Añadir columnas a una definición de tabla base.
- Crear o eliminar una clave primaria o una restricción de unicidad en una tabla base. Para obtener más información acerca de la autorización necesaria para crear o eliminar una clave primaria o una restricción de unicidad, consulte el apartado "ALTER TABLE" en la página 565.
- Crear o eliminar una clave foránea en una tabla base.  
También es necesario el privilegio REFERENCES en cada columna de la tabla padre.
- Crear o eliminar una restricción de comprobación en una tabla base.
- Crear un desencadenante en una tabla base.
- Añadir, restablecer o eliminar una opción de columna para un apodo.

## GRANT (privilegios de apodo, vista o tabla)

- Cambiar un nombre de columna de apodo o tipo de datos.
- Añadir o cambiar un comentario en una tabla base, vista o apodo.

### CONTROL

Otorga:

- Todos los privilegios adecuados de la lista, es decir:
  - ALTER, CONTROL, DELETE, INSERT, INDEX, REFERENCES, SELECT y UPDATE para tablas base
  - CONTROL, DELETE, INSERT, SELECT y UPDATE para vistas
  - ALTER, CONTROL, INDEX y REFERENCES para apodos
- La posibilidad de otorgarlos privilegios anteriores (excepto CONTROL) a otros.
- La posibilidad de eliminar la tabla base, vista o apodo.

Esta posibilidad no puede extenderse a otros sobre la base de poseer el privilegio CONTROL. La única manera que puede extenderse es otorgando el privilegio CONTROL en sí y esto sólo puede realizarlo alguien con la autorización SYSADM o DBADM.
- La posibilidad de ejecutar el programa de utilidad RUNSTATS en la tabla e índices. Consulte el manual *Consulta de mandatos* para obtener información acerca de RUNSTATS.
- La posibilidad de emitir SET CONSTRAINTS en la tabla base o la tabla de resumen.

La persona que define una tabla base, una tabla de resumen o un apodo recibe automáticamente el privilegio CONTROL.

La persona que define una vista recibe automáticamente el privilegio CONTROL si posee el privilegio CONTROL en todas las tablas, vistas y apodos identificados en la selección completa.

### DELETE

Otorga el privilegio para suprimir las filas de la tabla o vista actualizable.

### INDEX

Otorga el privilegio para crear un índice en una tabla o una especificación de índice en un apodo. Este privilegio no se puede otorgar en una vista. El creador de un índice o de una especificación de índice tiene automáticamente el privilegio CONTROL en el índice o en la especificación de índice (que autoriza al creador a eliminar el índice o la especificación de índice). Así mismo, el creador mantiene el privilegio CONTROL incluso si se revoca el privilegio INDEX.

### INSERT

Otorga el privilegio para insertar filas en la tabla o vista actualizable y para ejecutar el programa de utilidad IMPORT.

### REFERENCES

Otorga el privilegio para crear y eliminar una clave foránea que haga referencia a la tabla como la tabla padre.

Si el ID de autorización de la sentencia tiene uno de los privilegios siguientes:

- Autorización DBADM o SYSADM
- Privilegio CONTROL para la tabla
- REFERENCES WITH GRANT OPTION para la tabla

entonces los usuarios autorizados pueden crear restricciones de referencia utilizando como clave padre todas las columnas de la tabla, incluso las que se han añadido después mediante la sentencia ALTER TABLE. De lo contrario, los privilegios otorgados son todos los privilegios REFERENCES de columna otorgables que el ID de autorización de la sentencia tiene en la tabla identificada. Para obtener más información acerca de la autorización necesaria para crear o eliminar una clave foránea, consulte el apartado “ALTER TABLE” en la página 565.

Se puede otorgar el privilegio para un apodo aunque las claves foráneas no pueden definirse para apodos de referencia.

### REFERENCES (*nombre-columna,...*)

Otorga el privilegio para crear y eliminar una clave foránea utilizando solamente las columnas especificadas en la lista de columnas como clave padre. Cada *nombre-columna* debe ser un nombre no calificado que identifique una columna de la tabla identificada en la cláusula ON. El privilegio REFERENCES para columnas no puede otorgarse para tablas con tipo, vistas con tipo ni apodos (SQLSTATE 42997).

### SELECT

Otorga el privilegio para:

- Recupera filas de la tabla o vista.
- Crea vistas en la tabla.
- Ejecuta el programa de utilidad EXPORT en la tabla o vista. Consulte el manual *Consulta de mandatos* para obtener información acerca de EXPORT.

### UPDATE

Otorga el privilegio para utilizar la sentencia UPDATE sobre la tabla o vista actualizable identificada en la cláusula ON.

Si el ID de autorización de la sentencia tiene uno de los privilegios siguientes:

- Autorización DBADM o SYSADM
- El privilegio CONTROL en la tabla o vista
- UPDATE WITH GRANT OPTION en la tabla o vista

## GRANT (privilegios de apodo, vista o tabla)

entonces la persona o personas a las que se otorga pueden actualizar todas las columnas actualizables de la tabla o vista en las que la persona que otorga tiene el privilegio así como aquellas columnas que se han añadido después utilizando la sentencia ALTER TABLE. De lo contrario, los privilegios otorgados son los privilegios UPDATE de columna otorgables que el ID de autorización de la sentencia tiene en la tabla o vista identificada.

### UPDATE (*nombre-columna*,...)

Otorga el privilegio de utilizar la sentencia UPDATE para actualizar solamente las columnas especificadas en la lista de columnas. Cada *nombre-columna* debe ser un nombre no calificado que identifique una columna de la tabla o vista identificada en la cláusula ON. El privilegio UPDATE de nivel de columna no puede otorgarse en las tablas con tipo, vistas con tipo o apodos (SQLSTATE 42997).

### ON TABLE *nombre-tabla* o *nombre-vista* o *apodo*

Especifica la tabla, vista o apodo en la que se han de otorgar los privilegios.

No pueden otorgarse privilegios en una vista no operativa ni en una tabla de resumen no operativa (SQLSTATE 51024). No pueden otorgarse privilegios para una tabla temporal declarada (SQLSTATE 42995).

### TO

Especifica a quién se otorgan los privilegios.

### USER

Especifica que el *nombre-autorización* identifica a un usuario.

### GROUP

Especifica que el *nombre-autorización* identifica un nombre de grupo.

*nombre-autorización*,...

Lista los ID de autorización de uno o varios usuarios o grupos.<sup>100</sup>

Un privilegio otorgado a un grupo no se utiliza para el control de autorizaciones en las sentencias DML estáticas de un paquete. Tampoco se utiliza al comprobar la autorización en una tabla base cuando se procesa una sentencia CREATE VIEW.

En DB2 Universal Database, los privilegios de tabla otorgados a grupos sólo se aplican a las sentencias que se preparan dinámicamente. Por ejemplo, si se ha otorgado el privilegio INSERT en la tabla PROJECT al grupo D204 pero no a UBIQUITY (un miembro de D204), UBIQUITY podría emitir la sentencia:

```
EXEC SQL EXECUTE IMMEDIATE :INSERT_STRING;
```

---

100. Se han eliminado las restricciones que existían en versiones anteriores sobre las operaciones de otorgar los ID de autorización del usuario que emite la sentencia.



## GRANT (privilegios de apodo, vista o tabla)

en la que el contenido de la serie es:

```
INSERT INTO PROJECT (PROJNO, PROJNAME, DEPTNO, RESPEMP)
VALUES ('AD3114', 'TOOL PROGRAMMING', 'D21',
'000260');
```

pero no podría precompilar ni enlazar un programa con la sentencia:

```
EXEC SQL INSERT INTO PROJECT (PROJNO, PROJNAME,
DEPTNO, RESPEMP)
VALUES ('AD3114', 'TOOL PROGRAMMING', 'D21',
'000260');
```

### PUBLIC

Otorga los privilegios a todos los usuarios.<sup>101</sup>

### WITH GRANT OPTION

Permite que los *nombres-autorización* especificados otorguen (GRANT) los privilegios a otros.

Si los privilegios especificados incluyen CONTROL, WITH GRANT OPTION se aplica a todos los privilegios aplicables excepto CONTROL (SQLSTATE 01516).

## Normas

- Si no se especifica USER ni GROUP, entonces
  - Si se define el *nombre-autorización* en el sistema operativo sólo como GROUP, entonces se supone GROUP.
  - Si se define el *nombre-autorización* en el sistema operativo sólo como USER o si no está definido, se supone USER.
  - Si se define el *nombre-autorización* en el sistema operativo como ambos, o se utiliza la autenticación DCE, se genera un error (SQLSTATE 56092).
- En general, la sentencia GRANT procesará la operación de otorgar privilegios que el ID de autorización de la sentencia tenga permitido otorgar, devolviendo un aviso (SQLSTATE 01007) si no se han otorgado uno o varios privilegios. Si no se ha otorgado ningún privilegio, se devuelve un error (SQLSTATE 42501).<sup>102</sup> Si se especifica el privilegio CONTROL, los privilegios sólo serán otorgados si el ID de autorización de la sentencia tiene autoridad SYSADM o DBADM (SQLSTATE 42501).

## Notas

- Los privilegios se pueden otorgar independientemente a cada nivel de una jerarquía de tablas. Un usuario con un privilegio sobre una supertabla

---

101. Se han eliminado las restricciones que existían en las versiones anteriores sobre el uso de los privilegios otorgados a PUBLIC para las sentencias de SQL estáticas y las sentencias CREATE VIEW.

102. Si el paquete utilizado para procesar la sentencia se ha precompilado con LANGLEVEL establecido en SQL92E para MIA, se devuelve un aviso (SQLSTATE 01007) a menos que la persona que otorga NO tenga privilegios en el objeto de la operación de otorgar.

## GRANT (privilegios de apodo, vista o tabla)

puede afectar a las subtablas. Por ejemplo, una actualización que especifique la supertabla *T* puede mostrarse como un cambio en una fila en la subtabla *S* de *T* efectuado por un usuario con privilegio UPDATE sobre *T*, pero sin privilegio UPDATE sobre *S*. Un usuario sólo puede operar directamente en la subtabla si sostiene el privilegio necesario sobre la subtabla.

- Otorgar privilegios de apodo no tiene efecto sobre los privilegios de objeto de la fuente de datos (tabla o vista). Normalmente, la tabla o vista necesita privilegios de fuente de datos a los que un apodo hace referencia al intentar recuperar los datos.
- No se definen los privilegios DELETE, INSERT, SELECT y UPDATE para los apodos ya que las operaciones en los apodos dependen de los privilegios del ID de autorización utilizados en la fuente de datos cuando se procesa la sentencia que hace referencia el apodo.

### Ejemplos

*Ejemplo 1:* Otorgue todos los privilegios de la tabla WESTERN\_CR a PUBLIC.

```
GRANT ALL ON WESTERN_CR
TO PUBLIC
```

*Ejemplo 2:* Otorgue los privilegios adecuados de la tabla CALENDAR para que los usuarios PHIL y CLAIRE puedan leerla e insertar nuevas entradas en ella. No les permita cambiar ni eliminar ninguna de las entradas existentes.

```
GRANT SELECT, INSERT ON CALENDAR
TO USER PHIL, USER CLAIRE
```

*Ejemplo 3:* Otorgue todos los privilegios de la tabla COUNCIL al usuario FRANK y la posibilidad de extender todos los privilegios a otros.

```
GRANT ALL ON COUNCIL
TO USER FRANK WITH GRANT OPTION
```

*Ejemplo 4:* Otorgue (GRANT) el privilegio SELECT en la tabla CORPDATA.EMPLOYEE a un usuario llamado JOHN. Hay un usuario llamado JOHN y no hay ningún grupo llamado JOHN.

```
GRANT SELECT ON CORPDATA.EMPLOYEE TO JOHN
```

o

```
GRANT SELECT
ON CORPDATA.EMPLOYEE TO USER JOHN
```

*Ejemplo 5:* Otorgue (GRANT) el privilegio SELECT en la tabla CORPDATA.EMPLOYEE a un grupo llamado JOHN. Hay un grupo llamado JOHN y ningún usuario llamado JOHN.

```
GRANT SELECT ON CORPDATA.EMPLOYEE TO JOHN
```

## GRANT (privilegios de apodo, vista o tabla)

O

```
GRANT SELECT ON CORPDATA.EMPLOYEE TO GROUP JOHN
```

*Ejemplo 6:* Otorgue (GRANT) los privilegios INSERT y SELECT en la tabla T1 a un grupo llamado D024 y a un usuario llamado D024.

```
GRANT INSERT, SELECT ON TABLE T1
TO GROUP D024, USER D024
```

En este caso, tanto los miembros del grupo D024 como el usuario D024 tendrían permitido insertar (INSERT) y seleccionar (SELECT) en la tabla T1. También, se añadirían dos filas a la vista de catálogo SYSCAT.TABAUTH.

*Ejemplo 7:* Otorgue (GRANT) INSERT, SELECT y CONTROL en la tabla CALENDAR al usuario FRANK. FRANK debe poder pasar los privilegios a otros.

```
GRANT CONTROL ON TABLE CALENDAR
TO FRANK WITH GRANT OPTION
```

El resultado de esta sentencia es un aviso (SQLSTATE 01516) de que no se ha dado WITH GRANT OPTION a CONTROL. Frank tiene ahora la posibilidad de otorgar cualquier privilegio para CALENDAR, inclusive INSERT y SELECT como era necesario. FRANK no puede otorgar CONTROL en CALENDAR a otros usuarios a menos que tenga la autorización SYSADM o DBADM.

*Ejemplo 8:* El usuario JON ha creado un apodo para una tabla Oracle que no tiene índice. El apodo es ORAREM1. Más tarde, Oracle DBA ha definido un índice para esta tabla. El usuario SHAWN ahora desea que DB2 sepa que este índice existe, de modo que el optimizador pueda idear estrategias para acceder a la tabla de un modo más eficaz. SHAWN puede informar a DB2 del índice creando una especificación de índice para ORAREM1. Otorgue a SHAWN el privilegio para este apodo, de modo que podrá crear la especificación de índice.

```
GRANT INDEX ON NICKNAME ORAREM1
TO USER SHAWN
```

## GRANT (privilegios para espacios de tablas)

### GRANT (privilegios para espacios de tablas)

Esta forma de la sentencia GRANT otorga privilegios para un espacio de tablas.

#### Invocación

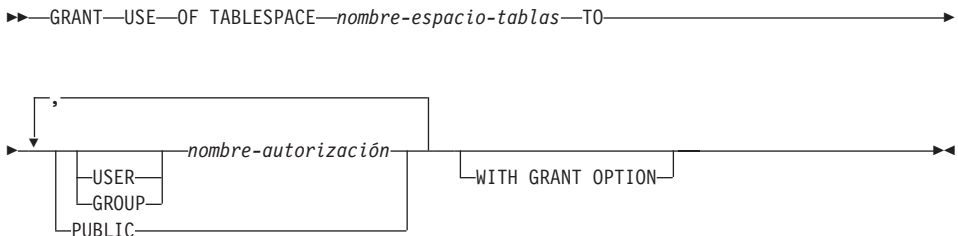
Esta sentencia se puede intercalar en un programa de aplicación o se puede emitir utilizando sentencias de SQL dinámicas. Es una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si se aplica la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

#### Autorización

El ID de autorización de la sentencia debe tener al menos uno de los siguientes privilegios:

- WITH GRANT OPTION para utilizar el espacio de tablas
- Autorización SYSADM, SYSCTRL o DBADM

#### Sintaxis



#### Descripción

##### USE

Otorga el privilegio para especificar, de forma explícita o por omisión, el espacio de tablas al crear una tabla. La opción GRANT otorga automáticamente el privilegio USE al creador de un espacio de tablas.

##### OF TABLESPACE *nombre-espacio-tablas*

Identifica el espacio de tablas para el que debe otorgarse el privilegio USE. El espacio de tablas no puede ser SYSCATSPACE (SQLSTATE 42838) ni un espacio de tablas temporal del sistema (SQLSTATE 42809).

##### TO

Especifica a quién se otorga el privilegio USE.

##### USER

Especifica que el *nombre-autorización* identifica a un usuario.

##### GROUP

Especifica que el *nombre-autorización* identifica un nombre de grupo.

## GRANT (privilegios para espacios de tablas)

### *nombre-autorización*

Lista los ID de autorización de uno o varios usuarios o grupos.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

### **PUBLIC**

Otorga el privilegio USE a todos los usuarios.

### **WITH GRANT OPTION**

Permite que el *nombre-autorización* especificado otorgue el privilegio USE a otros usuarios.

Si se omite WITH GRANT OPTION, el *nombre-autorización* especificado sólo puede otorgar el privilegio USE a otros usuarios si éstos:

- tienen autorización SYSADM o DBADM, o bien
- han recibido la capacidad de otorgar el privilegio USE desde alguna otra fuente.

### **Notas**

Si no se especifica USER ni GROUP, entonces

- Si se define el *nombre-autorización* en el sistema operativo sólo como GROUP, entonces se supone GROUP.
- Si el *nombre-autorización* está definido en el sistema operativo sólo como USER, o si no está definido, se supone USER.
- Si el *nombre-autorización* está definido en el sistema operativo como BOTH, o se utiliza la autenticación DCE, se produce un error (SQLSTATE 56092).

### **Ejemplos**

*Ejemplo 1:* Este ejemplo otorga al usuario BOBBY la capacidad para crear tablas en el espacio de tablas PLANS y para otorgar este privilegio a otros usuarios.

```
GRANT USE OF TABLESPACE PLANS TO BOBBY WITH GRANT OPTION
```

## INCLUDE

---

## INCLUDE

La sentencia INCLUDE inserta declaraciones en un programa fuente.

### Invocación

Esta sentencia sólo puede incluirse en un programa de aplicación. No es una sentencia ejecutable.

### Autorización

No se necesita.

### Sintaxis



### Descripción

#### SQLCA

Indica que se ha de incluir la descripción de un área de comunicaciones SQL (SQLCA). Para ver una descripción de la SQLCA, consulte el “Apéndice B. Comunicaciones SQL (SQLCA)” en la página 1261.

#### SQLDA

Indica que se ha de incluir la descripción de un área de descriptores SQL (SQLDA). Para ver una descripción de la SQLDA, consulte el “Apéndice C. Área de descriptores SQL (SQLDA)” en la página 1267.

#### *nombre*

Identifica un archivo externo que contiene el texto que se ha de incluir en el programa fuente que se está precompilando. Puede ser un identificador SQL sin ninguna extensión de nombre de archivo o un literal entre comillas simples ( ' ). Un identificador SQL asume la extensión del nombre de archivo del archivo fuente que se está precompilando. Si no se proporciona ninguna extensión de nombre de archivo mediante un literal entrecomillado, entonces no se asume ninguna.

Para obtener información específica del lenguaje principal, consulte el manual *Application Development Guide*.

### Notas

- Cuando se precompila un programa, la sentencia INCLUDE se sustituye por las sentencias fuente. Por lo tanto, la sentencia INCLUDE debe especificarse en un punto del programa en el que las sentencias fuente resultantes sean aceptables para el compilador.
- El archivo fuente externo debe estar escrito en el lenguaje principal especificado por el *nombre*. Si es superior a 18 caracteres o contiene caracteres que no están permitidos en un identificador SQL, debe ir

entrecorillado. Las sentencias INCLUDE *nombre* pueden estar anidadas pero no pueden ser cíclicas (por ejemplo, si A y B son módulos y A contiene una sentencia INCLUDE *nombre*, no es válido que A llame a B y, a continuación, que B llame a A).

- Si la opción de precompilación LANGLEVEL se establece en el valor SQL92E, no debe especificarse INCLUDE SQLCA. Las variables SQLSTATE y SQLCODE pueden estar definidas en la sección de declaraciones de variables de lenguaje principal.

## Ejemplo

Incluya una SQLCA en un programa C.

```
EXEC SQL INCLUDE SQLCA;

EXEC SQL DECLARE C1 CURSOR FOR
 SELECT DEPTNO, DEPTNAME, MGRNO FROM TDEPT
 WHERE ADMRDEPT = 'A00';

EXEC SQL OPEN C1;

mientras (SQLCODE==0) {
 EXEC SQL FETCH C1 INTO :dnum, :dname, :mnum;

 (Imprimir resultados)
}

EXEC SQL CLOSE C1;
```

# INSERT

---

## INSERT

La sentencia INSERT inserta filas en una tabla o vista. La inserción de una fila en una vista también inserta la fila en la tabla en la que se basa la vista.

### Invocación

Esta sentencia se puede intercalar en un programa de aplicación o se puede emitir utilizando sentencias de SQL dinámicas. Es una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

Para ejecutar esta sentencia, el ID de autorización de la sentencia debe tener como mínimo uno de los privilegios siguientes:

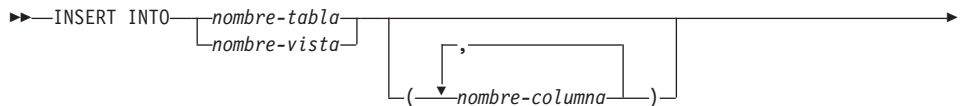
- Privilegio INSERT para la tabla o vista donde se deben insertar las filas
- Privilegio CONTROL para la tabla o vista donde se deben insertar las filas
- Autorización SYSADM o DBADM.

Además, para cada tabla o vista referenciada en una selección completa contenida en la sentencia INSERT, el ID de autorización de la sentencia debe tener como mínimo uno de los privilegios siguientes:

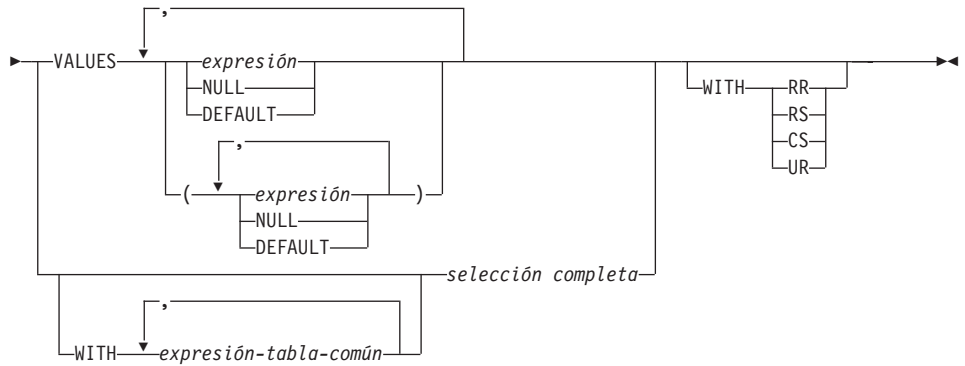
- Privilegio SELECT
- Privilegio CONTROL
- Autorización SYSADM o DBADM.

No se comprueban los privilegios GROUP para las sentencias INSERT estáticas.

### Sintaxis







**Nota:** En el “Capítulo 5. Consultas” en la página 471 se explica la sintaxis de *expresión-común-tabla* y *selección completa*.

## Descripción

**INTO** *nombre-tabla* o *nombre-vista*

Identifica el objeto de la operación de inserción. El nombre debe identificar una tabla o vista existente en el servidor de aplicaciones, pero no debe identificar una tabla de catálogo, una tabla de resumen, una vista de una tabla de catálogo ni una vista de sólo lectura.

No se puede insertar un valor en la columna de una vista que se derive de:

- Una constante, expresión o función escalar
- La misma columna de tabla base que otra columna de la vista.
- Una columna que se deriva de un apodo.

Si el objeto de la operación de inserción es una vista con dichas columnas, debe especificarse una lista de nombres de columna y dicha lista no debe identificar estas columnas.

**(nombre-columna,...)**

Especifica las columnas para las que se proporcionan valores de inserción. Cada nombre debe ser un nombre no calificado que identifica una columna de la tabla o vista. La misma columna no se puede identificar más de una vez. Una columna de vista que no puede aceptar valores de inserción no debe identificarse.

La omisión de la lista de columnas es una especificación implícita de una lista en la que cada columna de la tabla o vista se identifica en orden de izquierda a derecha. Esta lista se establece cuando se prepara la sentencia y, por lo tanto, no incluye las columnas que se han añadido a la tabla después de preparar la sentencia.

## INSERT

La lista de columnas implícita se establece en tiempo de preparación. Por lo tanto una sentencia INSERT incorporada en un programa de aplicación no utiliza ninguna columna que pueda haberse añadido a la tabla o vista después del tiempo de preparación.

### VALUES

Introduce una o varias filas de valores que se han de insertar.

Cada variable del lenguaje principal nombrada debe estar descrita en el programa de acuerdo con las reglas para la declaración de variables del lenguaje principal.

El número de valores para cada fila debe ser igual al número de nombres de la lista de columnas. El primer valor se inserta en la primera columna de la lista, el segundo valor en la segunda columna, etcétera.

### expresión

Una *expresión* puede ser tal como se define en el apartado “Expresiones” en la página 176.

### NULL

Especifica el valor nulo y sólo debe especificarse para las columnas con posibilidad de nulos.

### DEFAULT

Especifica que se ha de utilizar el valor por omisión. El resultado de especificar DEFAULT depende cómo se definió la columna, de este modo:

- Si la columna se definió como columna generada basándose en una expresión, el sistema genera el valor de la columna de acuerdo con esa expresión.
- Si se utiliza la cláusula IDENTITY, el gestor de bases de datos genera el valor.
- Si se utiliza la cláusula WITH DEFAULT, el valor insertado es tal como está definido para la columna (vea *cláusula-predefinida* en “CREATE TABLE” en la página 835).
- Si no se utilizan las cláusulas WITH DEFAULT, GENERATED ni NOT NULL, el valor insertado es NULL.
- Si se utiliza la cláusula NOT NULL y no la cláusula GENERATED, o no se utiliza WITH DEFAULT o se utiliza DEFAULT NULL, no se puede especificar la palabra clave DEFAULT para esa columna (SQLSTATE 23502).

### WITH *expresión-tabla-común*

Define una expresión de tabla común para utilizarla con la selección completa que va a continuación. Consulte el apartado “*expresión-común-tabla*” en la página 519 para ver una explicación de la *expresión-común-tabla*.

*selección completa*

Especifica un conjunto de filas nuevas en la forma de la tabla resultante de una selección completa. Puede haber una, más de una o ninguna. Si la tabla resultante está vacía, SQLCODE se establece en +100 y SQLSTATE se establece en '02000'.

Cuando el objeto base de INSERT y el objeto base de la selección completa o cualquier subconsulta de la selección completa, son la misma tabla, la selección completa se evalúa por completo antes de insertar alguna fila.

El número de columnas de la tabla resultante debe ser igual al número de nombres de la lista de columnas. El valor de la primera columna del resultado se inserta en la primera columna de la lista, el segundo valor en la segunda columna, etcétera.

**WITH**

Especifica el nivel de aislamiento en el que se ejecuta la selección completa (fullselect).

**RR**

Lectura repetible

**RS**

Estabilidad de lectura

**CS**

Estabilidad del cursor

**UR**

Lectura no confirmada

El nivel de aislamiento por omisión de la sentencia es el nivel de aislamiento del paquete en el que está enlazada la sentencia.

**Normas**

- **Valores por omisión:** El valor insertado en cualquier columna que no esté en la lista de columnas es el valor por omisión de la columna o nulo. Las columnas que no permiten valores nulos y no están definidas con NOT NULL WITH DEFAULT deben incluirse en la lista de columnas. De manera similar, si se inserta en una vista, el valor insertado en cualquier columna de la tabla base que no esté en la vista es el valor por omisión de la columna o nulo. De ahí que todas las columnas de la tabla base que no estén en la vista deben ser un valor por omisión o permitir valores nulos. El único valor que se puede insertar en una columna generada que se ha definido con la cláusula GENERATED ALWAYS es DEFAULT (SQLSTATE 428C9).
- **Longitud:** Si el valor de inserción de una columna es un número, la columna debe ser una columna numérica con la capacidad de representar la

## INSERT

parte entera del número. Si el valor de inserción de una columna es una serie, la columna debe ser una columna de serie con un atributo de longitud que como mínimo sea tan grande como la longitud de la serie, o una columna de fecha y hora si la serie representa una fecha, hora o indicación de la hora.

- **Asignación:** Los valores de inserción se asignan a columnas de acuerdo con las reglas de asignación descritas en el Capítulo 3.
- **Validez:** Si la tabla nombrada, o la tabla base de la vista nombrada, tiene uno o varios índices de unicidad, cada fila insertada en la tabla debe ajustarse a las restricciones impuestas por dichos índices. Si se nombra una vista cuya definición incluye WITH CHECK OPTION, cada fila insertada en la vista debe ajustarse a la definición de la vista. Para ver una explicación de las reglas que rigen esta situación, consulte el apartado “CREATE VIEW” en la página 957.
- **Integridad de referencia:** Para cada restricción definida en una tabla, cada valor de inserción que no sea nulo de la clave foránea debe ser igual al valor de clave primaria de la tabla padre.
- **Restricción de comprobación:** Los valores de inserción deben cumplir las condiciones de control de las restricciones de comprobación definidas en la tabla. En una sentencia INSERT para una tabla con restricciones de comprobación definidas, se evalúan las condiciones de restricción una vez para cada fila que se inserta.
- **Desencadenantes:** Las sentencias de inserción pueden provocar que se ejecuten los desencadenantes. Un desencadenante puede provocar que se ejecuten otras sentencias o puede generar condiciones de error basadas en los valores de inserción.
- **Enlaces de datos:** Las sentencias de inserción que incluyan valores DATALINK darán como resultado un intento de enlace del archivo si se incluye un valor de URL (ni una serie vacía ni espacios en blanco) y se define la columna con FILE LINK CONTROL. Los errores producidos en el valor DATALINK o en el enlace del archivo provocarán que falle la inserción (SQLSTATE 428D1 o 57050).

### Notas

- Después de la ejecución de una sentencia INSERT que está incorporada en un programa, el valor de la tercera variable de la parte de SQLERRD(3) de la SQLCA indica el número de filas que se han insertado. SQLERRD(5) contiene la cuenta de todas las operaciones de inserción, actualización y supresión activadas.
- A menos de que ya existan los bloqueos adecuados, se adquieren uno o varios bloqueos exclusivos durante la ejecución de una sentencia INSERT satisfactoria. Hasta que se liberen los bloqueos, sólo se puede acceder a una fila insertada:
  - El proceso de aplicación que ha realizado la inserción.

- Otro proceso de aplicación que utilice el nivel de aislamiento UR a través del cursor de sólo lectura, la sentencia SELECT INTO o la subselección utilizada en una subconsulta.
- Para obtener más información acerca del bloqueo, consulte la descripción de las sentencias COMMIT, ROLLBACK y LOCK TABLE.
- Si una aplicación se ejecuta en una base de datos particionada y se enlaza con la opción INSERT BUF, las sentencias INSERT con VALUES que no se procesan utilizando EXECUTE IMMEDIATE pueden ponerse en el almacenamiento intermedio. DB2 supone que tales sentencias INSERT se procesan dentro de un bucle de la lógica de la aplicación. En lugar de ejecutar la sentencia hasta que se completa, intenta almacenar los nuevos valores de fila en uno o varios almacenamientos intermedios. Como resultado las inserciones reales de las filas en la tabla se efectúan posteriormente, de manera asíncrona con la lógica de INSERT de la aplicación. Tenga en cuenta que esta inserción asíncrona puede generar un error relacionado con una sentencia INSERT que se devuelva a otra sentencia de SQL que siga a INSERT en la aplicación.  
 Esto tiene la posibilidad de mejorar significativamente el rendimiento de INSERT, pero se utiliza mejor con datos limpios, debido a la naturaleza asíncrona del manejo de errores. Consulte la inserción en almacenamiento intermedio en el manual *Application Development Guide* para obtener más detalles.
- Cuando se inserta una fila en una tabla que tiene una columna de identidad, DB2 genera un valor para la columna de identidad.
  - Para una columna de identidad definida como GENERATED ALWAYS, DB2 genera siempre el valor.
  - Para una columna definida como GENERATED BY DEFAULT, si no se especifica explícitamente un valor (con una cláusula VALUES o subselección), DB2 genera un valor.

El primer valor generado por DB2 es el valor de la especificación START WITH para la columna de identidad.

- Cuando se inserta un valor para una columna de identidad de un tipo diferenciado definido por el usuario, el cálculo completo se realiza en el tipo fuente, y el resultado se convierte al tipo diferenciado antes de asignar realmente el valor a la columna.<sup>103</sup>
- Cuando se inserta en una columna de identidad definida como GENERATED ALWAYS, DB2 genera siempre un valor para la columna, y el usuario no debe especificar un valor durante la inserción. Si una columna de identidad definida como GENERATED ALWAYS aparece en la lista de

---

103. Antes del cálculo no se realiza ninguna conversión del valor anterior al tipo fuente.

## INSERT

columnas de una sentencia INSERT, y la cláusula VALUES contiene un valor distinto del valor por omisión, se produce un error (SQLSTATE 428C9).

Por ejemplo, suponga que EMPID es una columna de identidad definida como GENERATED ALWAYS, entonces el mandato:

```
INSERT INTO T2 (EMPID, EMPNAME, EMPADDR)
VALUES (:hv_valid_emp_id, :hv_name, :hv_addr)
```

da como resultado un error.

- Cuando se inserta en una columna definida como GENERATED BY DEFAULT, DB2 permite especificar un valor real para la columna dentro de la cláusula VALUES o en una subselección. Sin embargo, cuando se especifica un valor en la cláusula VALUES, DB2 no realiza ninguna verificación del valor. Para asegurar la unicidad de los valores, se debe crear un índice de unicidad para la columna de identidad.

Cuando se inserta en una tabla que tiene una columna de identidad definida como GENERATED BY DEFAULT, y no se especifica una lista de columnas, la cláusula VALUES puede especificar la palabra clave DEFAULT para representar el valor de la columna de identidad. DB2 generará el valor para la columna de identidad.

```
INSERT INTO T2 (EMPID, EMPNAME, EMPADDR)
VALUES (DEFAULT, :hv_name, :hv_addr)
```

En este ejemplo, EMPID está definido como columna de identidad, y por tanto el valor insertado en esta columna es generado por DB2.

- La reglas para insertar en una columna de identidad utilizando una subselección son similares a las reglas para una inserción mediante una cláusula VALUES. Sólo se puede especificar un valor para una columna de identidad si ésta está definida como GENERATED BY DEFAULT.

Por ejemplo, suponga que T1 y T2 son tablas que tienen la misma definición, y ambas contienen las columnas *intcol1* e *identcol2* (las dos son de tipo INTEGER y la segunda columna tiene el atributo de identidad).

Considere la inserción siguiente:

```
INSERT INTO T2
SELECT *
FROM T1
```

Este ejemplo es conceptualmente equivalente a:

```
INSERT INTO T2 (intcol1,identcol2)
SELECT intcol1, identcol2
FROM T1
```

En ambos casos, la cláusula INSERT proporciona un valor explícito para la columna de identidad de T2. Esta especificación explícita puede dar un

valor para la columna de identidad, pero la columna de identidad de T2 debe estar definida como GENERATED BY DEFAULT. De lo contrario se produce un error (SQLSTATE 428C9).

Si una tabla tiene una columna definida como GENERATED ALWAYS, todavía es posible propagar todas las demás columnas de una tabla que tenga la misma definición. Por ejemplo, dadas las tablas T1 y T2 descritas anteriormente, se pueden propagar los valores intcol1 desde T1 a T2, mediante el SQL siguiente:

```
INSERT INTO T2 (intcol1)
SELECT intcol1
FROM T1
```

Observe que, debido a que identcol2 no está especificado en la lista de columnas, se le proporcionará su valor por omisión (generado).

- Cuando se inserta una fila en una tabla de una sola columna y ésta es una columna de identidad definida como GENERATED ALWAYS, es posible especificar un valor en VALUES mediante la palabra clave DEFAULT. En este caso, la aplicación no proporciona ningún valor para la tabla, y DB2 genera el valor para la columna de identidad.

```
INSERT INTO IDTABLE
VALUES(DEFAULT)
```

En la tabla del ejemplo anterior, formada por una sola columna que tiene el atributo de identidad, para insertar varias filas con una única sentencia INSERT puede utilizarse esta sentencia:

```
INSERT INTO IDTABLE
VALUES (DEFAULT), (DEFAULT), (DEFAULT), (DEFAULT)
```

- Cuando DB2 genera un valor para una columna de identidad, ese valor generado caduca; la próxima vez que sea necesario un valor, DB2 generará uno nuevo. Esto es válido aunque falle o se cancele una sentencia INSERT en la que interviene una columna de identidad.

Por ejemplo, suponga que se ha creado un índice de unicidad para la columna de identidad. Si al generar un valor para una columna de identidad se detecta una violación de clave duplicada, se produce un error (SQLSTATE 23505) y se considera que el valor generado para la columna de identidad ha caducado. Esto puede ocurrir si la columna de identidad está definida como GENERATED BY DEFAULT y el sistema intenta generar un nuevo valor, pero el usuario ha especificado explícitamente valores para la columna de identidad en sentencias INSERT anteriores. En este caso, el volver a emitir la misma sentencia INSERT puede producir un resultado satisfactorio. DB2 generará el valor siguiente para la columna de identidad y es posible que este valor siguiente sea exclusivo, y que la sentencia INSERT tenga éxito.

## INSERT

- Si al generar un valor para una columna de identidad se excede el valor máximo de la columna (o el valor mínimo en el caso de una secuencia descendente), se produce un error (SQLSTATE 23522). En este caso, el usuario debe eliminar la tabla y crear una nueva con una columna de identidad que tenga un rango mayor (es decir, cambiar el tipo de datos o valor de incremento de la columna para permitir un rango mayor de valores).

Por ejemplo, una columna de identidad puede haberse definido con el tipo de datos SMALLINT, y posteriormente agotarse los valores que se pueden asignar a la columna. Para redefinir la columna de identidad como INTEGER, es necesario descargar los datos, eliminar la tabla y volver a crearla con una nueva definición para la columna, y luego cargar los datos de nuevo. Cuando se redefine la tabla, es necesario especificar un valor START WITH para la columna de identidad, para el que próximo valor generado por DB2 sea el valor que sigue a continuación en la secuencia original. Para determinar el valor final, emita una consulta utilizando el valor MAX de la columna de identidad (para una secuencia ascendente) o el valor MIN (para una secuencia descendente), antes de descargar los datos.

### Ejemplos

*Ejemplo 1:* Inserte un nuevo departamento con las siguientes especificaciones en la tabla DEPARTMENT:

- El número de departamento (DEPTNO) es 'E31'
- El nombre de departamento (DEPTNAME) es 'ARCHITECTURE'
- Dirigido por (MGRNO) una persona con el número '00390'
- Informa al departamento (ADMRDEPT) 'E01'.

```
INSERT INTO DEPARTMENT
VALUES ('E31', 'ARCHITECTURE', '00390', 'E01')
```

*Ejemplo 2:* Inserte un nuevo departamento en la tabla DEPARTMENT como en el ejemplo 1, pero no asigne ningún director al nuevo departamento.

```
INSERT INTO DEPARTMENT (DEPTNO, DEPTNAME, ADMRDEPT)
VALUES ('E31', 'ARCHITECTURE', 'E01')
```

*Ejemplo 3:* Inserte dos nuevos departamentos utilizando una sentencia en la tabla DEPARTMENT como en el ejemplo 2, pero no asigne ningún director al nuevo departamento.

```
INSERT INTO DEPARTMENT (DEPTNO, DEPTNAME, ADMRDEPT)
VALUES ('B11', 'PURCHASING', 'B01'),
('E41', 'DATABASE ADMINISTRATION', 'E01')
```

*Ejemplo 4:* Cree una tabla temporal MA\_EMP\_ACT con las mismas columnas que la tabla EMP\_ACT. Cargue MA\_EMP\_ACT con las filas de la tabla EMP\_ACT con un nuevo número de proyecto (PROJNO) que empieza por las letras 'MA'.



```

CREATE TABLE MA_EMP_ACT
(EMPNO CHAR(6) NOT NULL,
 PROJNO CHAR(6) NOT NULL,
 ACTNO SMALLINT NOT NULL,
 EMPTIME DEC(5,2),
 EMSTDATE DATE,
 EMENDATE DATE)

INSERT INTO MA_EMP_ACT
SELECT * FROM EMP_ACT
WHERE SUBSTR(PROJNO, 1, 2) = 'MA'

```

*Ejemplo 5:* Utilice una sentencia del programa C para añadir un esqueleto de proyecto a la tabla PROJECT. Obtenga el número de proyecto (PROJNO), nombre de proyecto (PROJNAME), número de departamento (DEPTNO) y empleado responsable (RESPEMP) de las variables del lenguaje principal. Utilice la fecha actual como la fecha de inicio del proyecto (PRSTDATE). Asigne un valor NULL a las restantes columnas de la tabla.

```

EXEC SQL INSERT INTO PROJECT (PROJNO, PROJNAME, DEPTNO, RESPEMP, PRSTDATE)
VALUES (:PRJNO, :PRJNM, :DPTNO, :REMP, CURRENT DATE);

```

## LOCK TABLE

---

### LOCK TABLE

La sentencia LOCK TABLE impide que procesos de aplicación simultáneos cambien una tabla o impide que procesos de aplicación simultáneos utilicen una tabla.

#### Invocación

Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

#### Autorización

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Privilegio SELECT para la tabla
- Privilegio CONTROL para la tabla
- Autorización SYSADM o DBADM.

#### Sintaxis

```
→ LOCK TABLE nombre-tabla IN [SHARE | EXCLUSIVE] MODE →
```

#### Descripción

##### *nombre-tabla*

Identifica la tabla. El *nombre-tabla* debe identificar una tabla que exista en el servidor de aplicaciones, pero no debe identificar una tabla de catálogo. El nombre de tabla no puede ser un apodo (SQLSTATE 42809) ni una tabla temporal declarada (SQLSTATE 42995). Si *nombre-tabla* es una tabla con tipo, debe ser la tabla raíz de la jerarquía de tablas (SQLSTATE 428DR).

##### IN SHARE MODE

Impide que procesos de aplicación simultáneos ejecuten alguna operación que no sea de sólo lectura en la tabla.

##### IN EXCLUSIVE MODE

Impide a los procesos de aplicación simultáneos ejecutar cualquier operación en la tabla. Tenga en cuenta que EXCLUSIVE MODE no impide que los procesos de aplicación simultáneos que estén ejecutando en el nivel de aislamiento Lectura no confirmada (UR) ejecuten operaciones de sólo lectura en la tabla.

#### Notas

- El bloqueo se utiliza para evitar operaciones simultáneas. No se adquiere necesariamente un bloqueo durante la ejecución de la sentencia LOCK

TABLE si ya existe un bloqueo satisfactorio. El bloqueo que impide las operaciones simultáneas se conserva como mínimo hasta la terminación de la unidad de trabajo.

- En una base de datos particionada, primero se adquiere un bloqueo de tabla en la primera partición del grupo de nodos (la partición con el número más bajo) y después en otras particiones. Si se interrumpe la sentencia LOCK TABLE, la tabla puede estar bloqueada en algunas particiones y en otras no. Si ocurre esto, emita otra sentencia LOCK TABLE para completar el bloqueo de todas las particiones o emita una sentencia COMMIT o ROLLBACK para liberar los bloqueos actuales.
- Esta sentencia afecta a todas las particiones en grupo de nodos.

### **Ejemplo**

Obtenga un bloqueo en la tabla EMP. No permita que otros programas lean o actualicen la tabla.

```
LOCK TABLE EMP IN EXCLUSIVE MODE
```

## OPEN

---

## OPEN

La sentencia OPEN abre un cursor para que pueda utilizarse para leer filas de la tabla resultante.

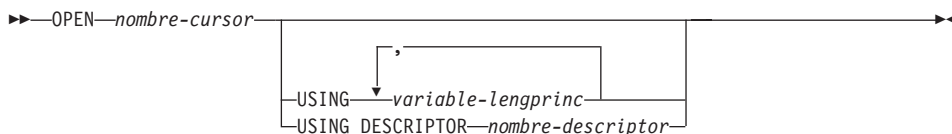
### Invocación

Aunque un recurso SQL interactivo pueda proporcionar una interfaz que dé la apariencia de ejecución interactiva, esta sentencia sólo puede incluirse dentro de un programa de aplicación. Se trata de una sentencia ejecutable que no puede prepararse de forma dinámica.

### Autorización

Consulte “DECLARE CURSOR” en la página 976 para conocer la autorización necesaria para utilizar un cursor.

### Sintaxis



### Descripción

*nombre-cursor*

Identifica un cursor que se define en una sentencia DECLARE CURSOR que se ha expresado antes en el programa. Cuando se ejecuta la sentencia OPEN, el cursor debe estar en el estado cerrado.

La sentencia DECLARE CURSOR debe identificar una sentencia SELECT, de una de las siguientes maneras:

- Incluyendo la sentencia SELECT en la sentencia DECLARE CURSOR
- Incluyendo un *nombre-sentencia* que identifique una sentencia SELECT preparada.

La tabla resultante del cursor se deriva evaluando dicha sentencia SELECT, utilizando los valores actuales de cualquier variable del lenguaje principal especificada en ella o en la cláusula USING de la sentencia OPEN. Las filas de la tabla resultante pueden derivarse durante la ejecución de la sentencia OPEN y puede crearse una tabla temporal para contenerlas; o pueden derivarse durante la ejecución de sentencias FETCH posteriores. En cualquier caso, el cursor se coloca en el estado abierto y se posiciona antes de la primera fila de su tabla resultante. Si la tabla está vacía, el estado del cursor está efectivamente “después de la última fila”.

### USING

Introduce una lista de variables del lenguaje principal cuyos valores se

sustituyen por los marcadores de parámetros (signos de interrogación) de una sentencia preparada. (Para obtener una explicación de marcadores de parámetros, consulte el apartado “PREPARE” en la página 1099.) Si la sentencia DECLARE CURSOR nombra una sentencia preparada que incluye marcadores de parámetros, debe utilizarse USING. Si la sentencia preparada no incluye ningún marcador de parámetros, se ignora USING.

*variable-length*

Identifica una variable descrita en el programa de acuerdo a las reglas para la declaración de variables del lenguaje principal. El número de variables debe ser igual al número de marcadores de parámetros de la sentencia preparada. La variable *n* corresponde al marcador de parámetro *n* de la sentencia preparada. Cuando sea adecuado, pueden proporcionarse variables localizadoras y variables de referencia a archivos como fuente de valores para marcadores de parámetros.

**DESCRIPTOR** *nombre-descriptor*

Identifica una SQLDA que debe contener una descripción válida de las variables del lenguaje principal.

Antes de procesar la sentencia OPEN, el usuario debe establecer los campos siguientes en la SQLDA:

- SQLN para indicar el número de apariciones de SQLVAR proporcionadas en la SQLDA
- SQLDABC para indicar el número de bytes de almacenamiento asignados para la SQLDA
- SQLD para indicar el número de variables utilizadas en la SQLDA al procesar la sentencia
- Las apariciones de SQLVAR, para indicar los atributos de las variables.

La SQLDA debe tener suficiente almacenamiento para contener todas las ocurrencias de SQLVAR. Por lo tanto, el valor de SQLDABC debe ser mayor o igual que  $16 + \text{SQLN} * (\text{N})$ , donde N es la longitud de una ocurrencia SQLVAR.

Si las columnas del resultado LOB necesitan acomodarse, debe haber dos entradas SQLVAR para cada elemento de la lista de selección (o columna de la tabla resultante). Consulte el apartado “Efecto de DESCRIBE en la SQLDA” en la página 1274, que explica las columnas SQLDOUBLED y LOB.

SQLD debe establecerse en un valor mayor o igual que cero y menor o igual que SQLN. Para obtener más información, vea “Apéndice C. Área de descriptores SQL (SQLDA)” en la página 1267.

## Normas

- Cuando se evalúa la sentencia SELECT del cursor, cada marcador de parámetros de la sentencia se sustituye efectivamente por su variable del lenguaje principal correspondiente. Para un marcador de parámetros con tipo, los atributos de la variable de destino son aquellos especificados por la especificación CAST. Para un marcador de parámetros sin tipo, los atributos de la variable de destino se determinan de acuerdo al contexto del marcador de parámetros. Consulte las reglas que afectan a los marcadores de parámetros en el apartado “Normas” en la página 1100.
- Supongamos que V indique una variable del lenguaje principal que corresponda al marcador de parámetros P. El valor de P se asigna a la variable de destino para P de acuerdo con las normas de asignación de un valor a una columna. Por lo tanto:
  - V debe ser compatible con el destino.
  - Si V es una serie, su longitud no debe ser mayor que el atributo de longitud del destino.
  - Si V es un número, el valor absoluto de su parte correspondiente al entero no debe ser mayor que el valor absoluto máximo de la parte correspondiente a los enteros del destino.
  - Si los atributos de V no son idénticos a los atributos del destino, el valor se convierte para ajustarse a los atributos del destino.

Cuando se evalúa la sentencia SELECT del cursor, el valor utilizado en lugar de P es el valor de la variable de destino para P. Por ejemplo, si V es CHAR(6) y el destino es CHAR(8), el valor que se utiliza en lugar de P es el valor de V rellenado con dos blancos.

- La cláusula USING está pensada para una sentencia SELECT preparada que contiene marcadores de parámetros. Sin embargo, también puede utilizarse cuando la sentencia SELECT del cursor forma parte de la sentencia DECLARE CURSOR. En este caso la sentencia OPEN se ejecuta como si cada variable del lenguaje principal de la sentencia SELECT fuese un marcador de parámetros, excepto que los atributos de las variables de destino son iguales a los atributos de las variables del lenguaje principal de una sentencia SELECT. El efecto es alterar temporalmente los valores de las variables del lenguaje principal de la sentencia SELECT del cursor con los valores de las variables del lenguaje principal especificadas en la cláusula USING.

## Notas

- **Estado cerrado de los cursores:** Todos los cursores de un programa están en estado cerrado cuando se inicia el programa y cuando éste inicia la sentencia ROLLBACK.

Todos los cursores, excepto los cursores abiertos declarados WITH HOLD, están en estado cerrado cuando el programa emite una sentencia COMMIT.

Un cursor también puede estar en estado cerrado debido a que se ha ejecutado una sentencia CLOSE o se ha detectado un error que ha originado que la posición del cursor fuese imprevisible.

- Para recuperar filas de la tabla resultante de un cursor, ejecute una sentencia FETCH cuando el cursor está abierto. La única manera de cambiar el estado de un cursor de cerrado a abierto es ejecutando la sentencia OPEN.
- *Efecto de tablas temporales:* En algunos casos, la tabla resultante de un cursor se deriva durante la ejecución de las sentencias FETCH. En otros casos, se utiliza en su lugar el método de tabla temporal. Con este método toda la tabla resultante se transfiere a una tabla temporal durante la ejecución de la sentencia OPEN. Cuando se utiliza una tabla temporal, los resultados de un programa pueden diferir de estas dos maneras:
  - Puede producirse un error durante OPEN que, de lo contrario, no ocurriría hasta alguna sentencia FETCH posterior.
  - Las sentencias INSERT, UPDATE y DELETE que se ejecutan en la misma transacción mientras el cursor está abierto no pueden afectar a la tabla resultante.

A la inversa, si no se utiliza una tabla temporal, las sentencias INSERT, UPDATE y DELETE ejecutadas mientras el cursor está abierto pueden afectar a la tabla resultante si se emiten desde la misma unidad de trabajo. El manual *Application Development Guide* describe cómo se puede utilizar el bloqueo para controlar el efecto de las operaciones INSERT, UPDATE y DELETE ejecutadas por unidades de trabajo simultáneas. La tabla resultante también puede verse afectada por operaciones ejecutadas en su propia unidad de trabajo y el efecto de dichas operaciones no es siempre previsible. Por ejemplo, si el cursor C está posicionado en una fila de su tabla resultante definida como SELECT \* FROM T y se inserta una nueva fila en T, el efecto de dicha inserción en la tabla resultante no es previsible ya que sus filas no están ordenadas. Por lo tanto, una sentencia FETCH C posterior puede recuperar la nueva fila de T o no.

- El poner en antememoria sentencias afecta a los cursores declarados abiertos por la sentencia OPEN. Consulte el apartado “Notas” en la página 1037 para obtener más información.

## Ejemplos

*Ejemplo 1:* Escriba las sentencias incorporadas en un programa COBOL que:

1. Definan un cursor C1 que se va a utilizar para recuperar todas las filas de la tabla DEPARTMENT para los departamentos que administra el departamento (ADMRDEPT) 'A00'.
2. Coloque el cursor C1 antes de la primera fila que se debe leer.

```
EXEC SQL DECLARE C1 CURSOR FOR
 SELECT DEPTNO, DEPTNAME, MGRNO
FROM DEPARTMENT
```

## OPEN

```
WHERE ADMRDEPT = 'A00'
END-EXEC.

EXEC SQL OPEN C1
END-EXEC.
```

*Ejemplo 2:* Codifique una sentencia OPEN para asociar un cursor DYN\_CURSOR con una sentencia de selección definida dinámicamente en un programa C. Suponiendo que se utilizan dos marcadores de parámetros en el predicado de la sentencia de selección, se suministran dos referencias a variables del lenguaje principal con la sentencia OPEN para pasar valores integer y varchar(64) entre la aplicación y la base de datos. (Las definiciones de variables del lenguaje principal relacionadas, la sentencia PREPARE y la sentencia DECLARE CURSOR también se muestran en el ejemplo siguiente.)

```
EXEC SQL BEGIN DECLARE SECTION;
 static short hv_int;
 char hv_vchar64[64];
 char stmt1_str[200];
EXEC SQL END DECLARE SECTION;

EXEC SQL PREPARE STMT1_NAME FROM :stmt1_str;
EXEC SQL DECLARE DYN_CURSOR CURSOR FOR STMT1_NAME;

EXEC SQL OPEN DYN_CURSOR USING :hv_int, :hv_vchar64;
```

*Ejemplo 3:* Codifique una sentencia OPEN en el ejemplo 2, pero en este caso el número de tipos de datos de los marcadores de parámetros de la cláusula WHERE no se conocen.

```
EXEC SQL BEGIN DECLARE SECTION;
 char stmt1_str[200];
EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLDA;

EXEC SQL PREPARE STMT1_NAME FROM :stmt1_str;
EXEC SQL DECLARE DYN_CURSOR CURSOR FOR STMT1_NAME;

EXEC SQL OPEN DYN_CURSOR USING DESCRIPTOR :sqlda;
```



## PREPARE

La sentencia PREPARE se utiliza por los programas de aplicación para preparar dinámicamente una sentencia de SQL para ejecución. La sentencia PREPARE crea una sentencia de SQL ejecutable, llamada una *sentencia preparada*, a partir de una forma de sentencia de serie de caracteres, denominada una *serie de sentencia*.

### Invocación

Esta sentencia sólo puede incluirse en un programa de aplicación. Se trata de una sentencia ejecutable que no puede prepararse de forma dinámica.

### Autorización

Para las sentencias en las que el control de autorizaciones se realiza al preparar la sentencia (DML), los privilegios del ID de autorización de la sentencia deben incluir los necesarios para ejecutar la sentencia de SQL especificada por la sentencia PREPARE. El ID de autorización de la sentencia puede verse afectado por la opción de enlace DYNAMICRULES. Consulte el apartado “Características del SQL dinámico en la ejecución” en la página 81.

Para las sentencias en las que el control de autorizaciones se realiza durante la ejecución (sentencias DDL, GRANT y REVOKE), no es necesaria ninguna autorización para utilizar la sentencia; no obstante, se comprueba la autorización cuando se ejecuta la sentencia preparada.

### Sintaxis

```

▶--PREPARE--nombre-sentencia-----▶
└-----INTO--nombre-descriptor-----┘

▶--FROM--variable-len princ-----▶

```

### Descripción

*nombre-sentencia*

Identifica la sentencia preparada. Si el nombre identifica una sentencia preparada existente, se destruye dicha sentencia preparada previamente. El nombre no debe identificar ninguna sentencia preparada que sea la sentencia SELECT de un cursor abierto.

**INTO**

Si se utiliza INTO y la sentencia PREPARE se ejecuta satisfactoriamente, la información acerca de la sentencia preparada se coloca en la SQLDA especificada por el nombre-descriptor.

## PREPARE

*nombre-descriptor*

Es el nombre de un SQLDA.<sup>104</sup>

### FROM

Introduce la serie de la sentencia. La serie de la sentencia es el valor de la variable del lenguaje principal especificada.

*variable-lengprinc*

Debe identificar una variable del lenguaje principal que esté descrita en el programa de acuerdo con las reglas para la declaración de variables de serie de caracteres. Debe ser una variable de serie de caracteres (de longitud fija o de longitud variable).

## Normas

- **Reglas para las sentencias:** La sentencia debe ser una sentencia ejecutable que se pueda preparar dinámicamente. Debe ser una de las siguientes sentencias de SQL:
  - ALTER
  - COMMENT ON
  - COMMIT
  - CREATE
  - DECLARE GLOBAL TEMPORARY TABLE
  - DELETE
  - DROP
  - EXPLAIN
  - FLUSH EVENT MONITOR
  - GRANT
  - INSERT
  - LOCK TABLE
  - REFRESH TABLE
  - RELEASE SAVEPOINT
  - RENAME TABLE
  - RENAME TABLESPACE
  - REVOKE
  - ROLLBACK
  - SAVEPOINT
  - sentencia-select
  - SET CURRENT DEFAULT TRANSFORM GROUP

---

104. La sentencia DESCRIBE puede utilizarse como una alternativa a esta cláusula. Consulte el apartado “DESCRIBE” en la página 997.

- SET CURRENT DEGREE
  - SET CURRENT EXPLAIN MODE
  - SET CURRENT EXPLAIN SNAPSHOT
  - SET CURRENT QUERY OPTIMIZATION
  - SET CURRENT REFRESH AGE
  - SET EVENT MONITOR STATE
  - SET INTEGRITY
  - SET PASSTHRU
  - SET PATH
  - SET SCHEMA
  - SET SERVER OPTION
  - UPDATE
- *Marcadores de parámetros:* Aunque una serie de sentencia no puede incluir referencias a variables del lenguaje principal, puede incluir *marcadores de parámetros*. Estos pueden sustituirse por los valores de las variables del lenguaje principal cuando se ejecuta la sentencia preparada. Un marcador de parámetros es un signo de interrogación (?) que se declara donde podría especificarse una variable del lenguaje principal si la serie de sentencia fuese una sentencia de SQL estática. Para ver una explicación de cómo se sustituyen por valores los marcadores de parámetros, consulte el apartado "OPEN" en la página 1094 y el apartado "EXECUTE" en la página 1036 .

Existen dos tipos de marcadores de parámetros:

#### *Marcador de parámetros con tipo*

Es un marcador de parámetros que se especifica junto con su tipo de datos de destino. Tiene el formato general:

```
CAST(? AS tipo de datos)
```

Esta notación no es una llamada a función, sino una "promesa" de que el tipo del parámetro en tiempo de ejecución será el tipo de datos especificado o algún otro tipo de datos que pueda convertirse al tipo de datos especificado. Por ejemplo, en:

```
UPDATE EMPLOYEE
SET LASTNAME = TRANSLATE(CAST(? AS VARCHAR(12)))
WHERE EMPNO = ?
```

el valor del argumento de la función TRANSLATE se proporcionará en tiempo de ejecución. El tipo de datos de dicho valor será VARCHAR(12) o algún tipo que pueda convertirse a VARCHAR(12).

#### *Marcador de parámetros sin tipo*

Es un marcador de parámetros que se especifica sin su tipo de datos de destino. Consta de un signo de interrogación individual. El tipo de

## PREPARE

datos de un marcador de parámetros sin tipo se proporciona por el contexto. Por ejemplo, el marcador de parámetros sin tipo del predicado de la sentencia de actualización anterior es el mismo que el tipo de datos de la columna EMPNO.

Los marcadores de parámetros con tipo pueden utilizarse en sentencias de SQL dinámicas donde esté soportada una variable del lenguaje principal y el tipo de datos se base en la promesa realizada en la función CAST.

Los marcadores de parámetros sin tiempo pueden utilizarse en sentencias de SQL dinámicas en las ubicaciones seleccionadas donde estén soportadas las variables del lenguaje principal. Estas ubicaciones y el tipo de datos resultante pueden encontrarse en la Tabla 30. Las ubicaciones se agrupan en esta tabla en expresiones, predicados y funciones para ayudar a determinar la posibilidad de aplicación de un marcador de parámetros sin tipo. Cuando se utiliza un marcador de parámetros sin tipo en una función (incluyendo operadores aritméticos, operadores CONCAT y de fecha y hora) con un nombre de función no calificado, el calificador se establece en 'SYSIBM' con el propósito de la resolución de la función.

Tabla 30. Uso del marcador de parámetros sin tipo

| Ubicación del marcador de parámetros sin tipo                                                                                                                                         | Tipo de datos                       |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------|
| <b>Expresiones (inclusive la lista de selección, CASE y VALUES)</b>                                                                                                                   |                                     |
| Solo en una lista de selección                                                                                                                                                        | Error                               |
| Ambos operandos de un solo operador aritmético, después de considerar la prioridad de los operadores y el orden de las reglas de la operación.                                        | Error                               |
| Incluye casos como:<br>? + ? + 10                                                                                                                                                     |                                     |
| Un operando de un solo operador de una expresión aritmética (no una expresión de fecha y hora)                                                                                        | El tipo de datos del otro operando. |
| Incluye casos como:<br>? + ? * 10                                                                                                                                                     |                                     |
| Duración etiquetada dentro de una expresión de fecha y hora. (Observe que la parte de una duración etiquetada que indica el tipo de unidades no puede ser un marcador de parámetros.) | DECIMAL(15,0)                       |

Tabla 30. Uso del marcador de parámetros sin tipo (continuación)

| Ubicación del marcador de parámetros sin tipo                                                                                                                                                    | Tipo de datos                                                                                                                                                                                                                                                                                                                                                        |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Cualquier otro operando de una expresión de fecha y hora (por ejemplo 'timecol + ?' o '? - datecol').                                                                                            | Error                                                                                                                                                                                                                                                                                                                                                                |
| Ambos operandos de un operador CONCAT                                                                                                                                                            | Error                                                                                                                                                                                                                                                                                                                                                                |
| Un operando de un operador CONCAT en que el otro operando es un tipo de datos de caracteres que no es CLOB                                                                                       | Si un operando es CHAR(n) o VARCHAR(n), donde n es menor que 128, entonces el otro es VARCHAR(254 - n). En todos los demás casos el tipo de datos es VARCHAR(254).                                                                                                                                                                                                   |
| Un operando de un operador CONCAT en que el otro operando es un tipo de datos gráfico que no es DBCLOB.                                                                                          | Si un operando es GRAPHIC(n) o VARGRAPHIC(n), donde n es menor que 64, entonces el otro es VARCHAR(127 - n). En todos los demás casos el tipo de datos es VARCHAR(127).                                                                                                                                                                                              |
| Un operando de un operador CONCAT en que el otro operando es una serie de gran objeto.                                                                                                           | Igual que el del otro operando.                                                                                                                                                                                                                                                                                                                                      |
| Como un valor a la derecha de una cláusula SET de una sentencia UPDATE.                                                                                                                          | El tipo de datos de la columna. Si la columna se define como un tipo diferenciado definido por el usuario, es el tipo de datos fuente del tipo diferenciado definido por el usuario. Si la columna está definida como tipo diferenciado definido por el usuario, es el tipo de datos estructurado y también indica el tipo devuelto de la función de transformación. |
| La expresión que está a continuación de la palabra clave CASE en una expresión CASE simple                                                                                                       | Error                                                                                                                                                                                                                                                                                                                                                                |
| Como mínimo una de las expresiones-resultado en una expresión CASE (tanto Simple como Con búsqueda) con el resto de las expresiones-resultado que sean marcadores de parámetros sin tipo o NULL. | Error                                                                                                                                                                                                                                                                                                                                                                |
| Cualquiera o todas las expresiones que siguen a WHEN en una expresión CASE simple.                                                                                                               | El resultado de aplicar el apartado "Reglas para los tipos de datos del resultado" en la página 120 a la expresión que sigue a CASE y las expresiones que siguen a WHEN que no son marcadores de parámetros sin tipo.                                                                                                                                                |

Tabla 30. Uso del marcador de parámetros sin tipo (continuación)

| Ubicación del marcador de parámetros sin tipo                                                                                                                                                                                                              | Tipo de datos                                                                                                                                                                                                                                                                                                                                                        |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Una expresión-resultado de una expresión CASE (tanto Simple como Con búsqueda) en la que por lo menos una expresión-resultado no es NULL y tampoco un marcador de parámetros sin tipo.                                                                     | Resultado de aplicar el apartado Reglas para los tipos de datos del resultado a todas las expresiones-resultado que no sean NULL ni marcadores de parámetros sin tipo.                                                                                                                                                                                               |
| Solo como una expresión-columna en una cláusula VALUES de una sola fila que no está en la sentencia INSERT.                                                                                                                                                | Error                                                                                                                                                                                                                                                                                                                                                                |
| Solo como una expresión-columna en una cláusula VALUES de múltiples filas que no está dentro de una sentencia INSERT y para la que las expresiones-columna de la misma posición de todas las demás expresiones-fila son marcadores de parámetros sin tipo. | Error                                                                                                                                                                                                                                                                                                                                                                |
| Solo como una expresión-columna en una cláusula VALUES de múltiples filas que no está dentro de una sentencia INSERT y para la que la expresión de la misma posición de como mínimo otra expresión-fila no es un marcador de parámetros sin tipo ni NULL.  | Resultado de aplicar el apartado “Reglas para los tipos de datos del resultado” en la página 120 en todos los operandos que no son marcadores de parámetros sin tipo.                                                                                                                                                                                                |
| Solo como una expresión-columna en una cláusula VALUES de una sola fila dentro de una sentencia INSERT.                                                                                                                                                    | El tipo de datos de la columna. Si la columna se define como un tipo diferenciado definido por el usuario, es el tipo de datos fuente del tipo diferenciado definido por el usuario. Si la columna está definida como tipo diferenciado definido por el usuario, es el tipo de datos estructurado y también indica el tipo devuelto de la función de transformación. |
| Solo como una expresión-columna en una cláusula VALUES de múltiples filas dentro de una sentencia INSERT.                                                                                                                                                  | El tipo de datos de la columna. Si la columna se define como un tipo diferenciado definido por el usuario, es el tipo de datos fuente del tipo diferenciado definido por el usuario. Si la columna está definida como tipo diferenciado definido por el usuario, es el tipo de datos estructurado y también indica el tipo devuelto de la función de transformación. |
| Como un valor a la derecha de una sentencia SET de registro especial                                                                                                                                                                                       | El tipo de datos del registro especial.                                                                                                                                                                                                                                                                                                                              |

Tabla 30. Uso del marcador de parámetros sin tipo (continuación)

| Ubicación del marcador de parámetros sin tipo                                                               | Tipo de datos                                                                                                                                                                                                     |
|-------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Predicados</b>                                                                                           |                                                                                                                                                                                                                   |
| Ambos operandos de un operador de comparación                                                               | Error                                                                                                                                                                                                             |
| Un operando de un operador de comparación en que el otro operando no es un marcador de parámetros sin tipo. | El tipo de datos del otro operando.                                                                                                                                                                               |
| Todos los operandos del predicado BETWEEN                                                                   | Error                                                                                                                                                                                                             |
| Ó<br>1o y 2o ó<br>1o y 3o<br><br>operandos de un predicado BETWEEN                                          | Igual que el del único marcador sin parámetros.                                                                                                                                                                   |
| Restantes situaciones BETWEEN (p. ej., un marcador de parámetros sin tipo solamente)                        | Resultado de aplicar el apartado “Reglas para los tipos de datos del resultado” en la página 120 en todos los operandos que no son marcadores de parámetros sin tipo.                                             |
| Todos los operandos de un predicado IN                                                                      | Error                                                                                                                                                                                                             |
| Los operandos 1 y 2 de un predicado IN.                                                                     | Resultado de aplicar el apartado Reglas para los tipos de datos del resultado en todos los operandos de la lista IN (operandos a la derecha de la palabra clave IN) que no son marcadores de parámetros sin tipo. |
| El primer operando de un predicado IN donde el lado derecho es una selección completa.                      | El tipo de datos de la columna seleccionada                                                                                                                                                                       |
| Cualquiera o todos los operandos de la lista IN del predicado IN                                            | Resultado de aplicar las reglas sobre tipos de datos resultantes en todos los operandos del predicado IN (operandos a izquierda y derecha del predicado IN) que no sean marcadores de parámetros sin tipo.        |
| El primer operando y cero o más operandos de la lista IN excluyendo el primer operando de la lista IN       | Resultado de aplicar el apartado Reglas para los tipos de datos del resultado en todos los operandos de la lista IN (operandos a la derecha de la palabra clave IN) que no son marcadores de parámetros sin tipo. |

*Tabla 30. Uso del marcador de parámetros sin tipo (continuación)*

| <b>Ubicación del marcador de parámetros sin tipo</b>                                                                                                                    | <b>Tipo de datos</b>                                                                                                                                                                                                                                                       |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Los tres operandos del predicado LIKE.                                                                                                                                  | La expresión coincidente (operando 1) y la expresión patrón (operando 2) son VARCHAR(32672). La expresión de escape (operando 3) es VARCHAR(2).                                                                                                                            |
| La expresión coincidente del predicado LIKE cuando la expresión patrón o la expresión de escape no son un marcador de parámetros sin tipo.                              | VARCHAR(32672) o VARGRAPHIC(16336) en función del tipo de datos del primer operando que no es un marcador de parámetros sin tipo.                                                                                                                                          |
| La expresión patrón del predicado LIKE cuando la expresión coincidente o la expresión de escape no es un marcador de parámetros sin tipo. Vea “Notas” en la página 227. | VARCHAR(32672) o VARGRAPHIC(16336) en función del tipo de datos del primer operando que no es un marcador de parámetros sin tipo. Si el tipo de datos de la expresión coincidente es BLOB, se supone que el tipo de datos de la expresión patrón es BLOB(32672).           |
| La expresión de escape del predicado LIKE cuando la expresión coincidente o la expresión patrón no es un marcador de parámetros sin tipo.                               | VARCHAR(2) o VARGRAPHIC(1) dependiendo del tipo de datos del primer operando que no es un marcador de parámetros sin tipo. Si el tipo de datos de la expresión coincidente o de la expresión patrón es BLOB, se supone el tipo de datos de la expresión de escape BLOB(1). |
| Operando del predicado NULL                                                                                                                                             | error                                                                                                                                                                                                                                                                      |
| <b>Funciones</b>                                                                                                                                                        |                                                                                                                                                                                                                                                                            |
| Todos los operandos de COALESCE (también llamados VALUE) o NULLIF                                                                                                       | Error                                                                                                                                                                                                                                                                      |
| Cualquier operando de COALESCE o NULLIF donde al menos el primer operando sea distinto de un marcador de parámetros sin tipo.                                           | Resultado de aplicar el apartado “Reglas para los tipos de datos del resultado” en la página 120 en todos los operandos que no son marcadores de parámetros sin tipo.                                                                                                      |
| POSSTR (ambos operandos)                                                                                                                                                | Ambos operandos son VARCHAR(32672).                                                                                                                                                                                                                                        |
| POSSTR (un operando en que el otro operando es de un tipo de datos de caracteres).                                                                                      | VARCHAR(32672).                                                                                                                                                                                                                                                            |
| POSSTR (un operando en que el otro operando es de un tipo de datos gráficos).                                                                                           | VARGRAPHIC(16336).                                                                                                                                                                                                                                                         |
| POSSTR (el operando de serie-búsqueda cuando el otro operando es BLOB).                                                                                                 | BLOB(32672).                                                                                                                                                                                                                                                               |
| SUBSTR (operando 1)                                                                                                                                                     | VARCHAR(32672)                                                                                                                                                                                                                                                             |
| SUBSTR (operandos 2 y 3)                                                                                                                                                | INTEGER                                                                                                                                                                                                                                                                    |



Tabla 30. Uso del marcador de parámetros sin tipo (continuación)

| Ubicación del marcador de parámetros sin tipo                                                                     | Tipo de datos                                                                                                                 |
|-------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| El operando 1 de la función escalar TRANSLATE.                                                                    | Error                                                                                                                         |
| Los operandos 2 y 3 de la función escalar TRANSLATE.                                                              | VARCHAR(32672) si el primer operando es un tipo de caracteres.<br>VARGRAPHIC(16336) si el primer operando es un tipo gráfico. |
| El operando 4 de la función escalar TRANSLATE.                                                                    | VARCHAR(1) si el primer operando es un tipo de caracteres. VARGRAPHIC(1) si el primer operando es un tipo gráfico.            |
| El segundo operando de la función escalar TIMESTAMP.                                                              | TIME                                                                                                                          |
| Unary minus                                                                                                       | DOUBLE-PRECISION                                                                                                              |
| Unary plus                                                                                                        | DOUBLE-PRECISION                                                                                                              |
| Todos los demás operandos de las restantes funciones escalares incluyendo las funciones definidas por el usuario. | Error                                                                                                                         |
| Operando de una función de columna                                                                                | Error                                                                                                                         |

## Notas

- Cuando se ejecuta una sentencia PREPARE, la serie de sentencia se analiza y se comprueba si hay errores. Si la sentencia no es válida, la condición de error se notifica en la SQLCA. Cualquier sentencia EXECUTE u OPEN que hace referencia a esta sentencia también recibirá el mismo error (debido a una preparación implícita realizada por el sistema) a menos que se haya corregido el error.
- Se puede hacer referencia a sentencias preparadas en las siguientes clases de sentencias, con las restricciones que se indican:

| En...                 | La sentencia preparada ... |
|-----------------------|----------------------------|
| <b>DECLARE CURSOR</b> | debe ser SELECT            |
| <b>EXECUTE</b>        | <i>no</i> debe ser SELECT  |

- Una sentencia preparada se puede ejecutar muchas veces. En efecto, si una sentencia preparada no se ejecuta más de una vez y no contiene marcadores de parámetros, es más eficaz utilizar la sentencia EXECUTE IMMEDIATE en lugar de las sentencias PREPARE y EXECUTE.
- La puesta en antememoria de la sentencia afecta a las preparaciones repetidas. Consulte el apartado "Notas" en la página 1037 para obtener más información.

## PREPARE

- Consulte el manual *Application Development Guide* para ver ejemplos de sentencias de SQL dinámicas de los lenguajes principales soportados.

### Ejemplos

*Ejemplo 1:* Prepare y ejecute una sentencia que no es de selección en un programa COBOL. Suponga que la sentencia está contenida en una variable del lenguaje principal HOLDER y que el programa sustituirá una serie de sentencia de una variable del lenguaje principal basada en algunas instrucciones del usuario. La sentencia que se ha de preparar no tiene ningún marcador de parámetros.

```
EXEC SQL PREPARE STMT_NAME FROM :HOLDER
END-EXEC.
```

```
EXEC SQL EXECUTE STMT_NAME
END-EXEC.
```

*Ejemplo 2:* Prepare y ejecute una sentencia que no es de selección como en el ejemplo 1, excepto codificada para un programa C. Suponga también que la sentencia que se va a preparar puede contener cualquier cantidad de marcadores de parámetros.

```
EXEC SQL PREPARE STMT_NAME FROM :holder;
EXEC SQL EXECUTE STMT_NAME USING DESCRIPTOR :insert_da;
```

Suponga que se ha de preparar la sentencia siguiente:

```
INSERT INTO DEPT VALUES(?, ?, ?, ?)
```

Las columnas de la tabla DEPT se definen de la siguiente manera:

```
DEPT_NO CHAR(3) NOT NULL, -- número de departamento
DEPTNAME VARCHAR(29), -- nombre de departamento
MGRNO CHAR(6), -- número de director
ADMNDEPT CHAR(3) -- número de departamento de administración
```

|         |     |              |
|---------|-----|--------------|
| SQLDAID | 192 |              |
| SQLDABC | 4   |              |
| SQLN    | 4   |              |
| SQLD    |     |              |
| SQLTYPE | 452 |              |
| SQLLEN  | 3   |              |
| SQLDATA |     | → G01        |
| SQLIND  |     |              |
| SQLNAME |     |              |
| SQLTYPE | 449 |              |
| SQLLEN  | 29  |              |
| SQLDATA |     | → COMPLAINTS |
| SQLIND  |     | → 0          |
| SQLNAME |     |              |
| SQLTYPE | 453 |              |
| SQLLEN  | 6   |              |
| SQLDATA |     |              |
| SQLIND  |     | → -1         |
| SQLNAME |     |              |
| SQLTYPE | 453 |              |
| SQLLEN  | 3   |              |
| SQLDATA |     | → A00        |
| SQLIND  |     | → 0          |
| SQLNAME |     |              |

Para insertar el número de departamento G01 llamado COMPLAINTS, que no tiene ningún director y que informa al departamento A00, la estructura INSERT\_DA debe tener los siguientes de más arriba antes de emitir la sentencia EXECUTE.

## REFRESH TABLE

---

### REFRESH TABLE

La sentencia REFRESH TABLE renueva los datos de una tabla de resumen.

#### Invocación

Esta sentencia se puede intercalar en un programa de aplicación o se puede emitir utilizando sentencias de SQL dinámicas. Es una sentencia ejecutable que puede prepararse de forma dinámica.

#### Autorización

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Privilegio CONTROL para la tabla.

#### Sintaxis

```
REFRESH TABLE nombre-tabla
```

#### Descripción

*nombre-tabla*

Identifica la tabla que se debe renovar.

El nombre, incluyendo el esquema implícito o explícito, debe identificar una tabla que ya exista en el servidor actual. La tabla debe permitir la sentencia REFRESH TABLE (SQLSTATE 42809). Incluye tablas de resumen definidas con:

- REFRESH IMMEDIATE
- REFRESH DEFERRED

## RELEASE (Conexión)

Esta sentencia coloca una o más conexiones en el estado de pendiente de liberación.

### Invocación

Aunque un recurso SQL interactivo pueda proporcionar una interfaz que dé la apariencia de ejecución interactiva, esta sentencia sólo puede incluirse dentro de un programa de aplicación. Se trata de una sentencia ejecutable que no puede prepararse de forma dinámica.

### Autorización

No se necesita ninguna.

### Sintaxis



### Notas:

- 1 Observe que un servidor de aplicaciones llamado `CURRENT` o `ALL` sólo puede identificarse mediante una variable del lenguaje principal.

### Descripción

*nombre-servidor* o *variable-lengprinc*

Identifica el servidor de aplicaciones mediante el *nombre-servidor* especificado o una *variable-lengprinc* que contenga el *nombre-servidor*.

Si se especifica una *variable-lengprinc*, debe ser una variable de serie de caracteres con un atributo de longitud no superior a 8 y no debe contener una variable indicadora. El *nombre-servidor* que está contenido en la *variable-lengprinc* debe estar justificado por la izquierda y no debe estar delimitado por comillas.

Observe que el *nombre-servidor* es un seudónimo de base de datos que identifica al servidor de aplicaciones. Debe estar listado en el directorio local del petionario de aplicaciones.

El seudónimo-basedatos especificado o el seudónimo-basedatos contenido en la variable del lenguaje principal debe identificar una conexión existente del proceso de aplicación. Si el seudónimo-basedatos no identifica ninguna conexión existente, se genera un error (SQLSTATE 08003).

## RELEASE (Conexión)

### CURRENT

Identifica la conexión actual del proceso de aplicación. El proceso de aplicación debe estar en el estado conectado. Si no se genera un error (SQLSTATE 08003).

### ALL

Indica que todas las conexiones existentes del proceso de aplicación. Este formato de la sentencia RELEASE coloca todas las conexiones existentes del proceso de aplicación en el estado pendiente de liberación. Por lo tanto, todas las conexiones se destruirán durante la siguiente operación de confirmación. No se produce ningún error ni aviso si no existen conexiones cuando se ejecuta la sentencia. La palabra clave opcional SQL se incluye para ser compatible con la sintaxis DB2/MVS SQL.

## Notas

### Ejemplos

*Ejemplo 1:* La aplicación ya no necesita la conexión SQL con IBMSTHDB. La sentencia siguiente hará que se destruya durante la siguiente operación de confirmación:

```
EXEC SQL RELEASE IBMSTHDB;
```

*Ejemplo 2:* La aplicación ya no necesita la conexión actual. La sentencia siguiente hará que se destruya durante la siguiente operación de confirmación:

```
EXEC SQL RELEASE CURRENT;
```

*Ejemplo 3:* Si una aplicación no tiene necesidad de acceder a las bases de datos después de una confirmación pero va a continuar en ejecución durante un período de tiempo, es mejor no unir dichas conexiones innecesariamente. La sentencia siguiente puede ejecutarse antes de la confirmación para asegurar que todas las conexiones se destruyan en la confirmación:

```
EXEC SQL RELEASE ALL;
```

## RELEASE SAVEPOINT

La sentencia RELEASE SAVEPOINT sirve para indicar que la aplicación ya no desea mantener el punto de salvar especificado. Después de invocar esta sentencia, ya no es posible hacer una retrotracción hasta el punto de salvar.

### Invocación

Esta sentencia puede incluirse en un programa de aplicación o emitirse de forma interactiva. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

No se necesita.

### Sintaxis

```

▶▶—RELEASE—TO—SAVEPOINT—nombre-puntosalvar—▶▶

```

### Descripción

*nombre-puntosalvar*

Se libera el punto de salvar especificado. Ya no es posible hacer una retrotracción hasta ese punto de salvar. Si el nombre del punto de salvar no existe, se devuelve un error (SQLSTATE 3B001).

### Notas

- El nombre del punto de salvar que se liberó se puede volver a utilizar en otra sentencia SAVEPOINT, aunque se haya especificado la palabra clave UNIQUE en una sentencia SAVEPOINT anterior donde se utiliza este mismo nombre de punto de salvar.

### Ejemplo

*Ejemplo 1:* Liberación de un punto de salvar llamado SAVEPOINT1.

```
RELEASE SAVEPOINT SAVEPOINT1
```

## RENAME TABLE

---

### RENAME TABLE

La sentencia RENAME TABLE cambia el nombre de una tabla existente.

#### Invocación

Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que se puede preparar dinámicamente. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

#### Autorización

Los privilegios que posee el ID de autorización de la sentencia deben incluir la autorización SYSADM o DBADM o el privilegio CONTROL.

#### Sintaxis

```
➤➤—RENAME—TABLE—nombre-tabla—TO—identificador-tabla-nuevo—➤➤
```

#### Descripción

##### *nombre-tabla*

Identifica la tabla existente que se ha de red denominar. El nombre, incluyendo el nombre de esquema, debe identificar una tabla que ya exista en la base de datos (SQLSTATE 42704). Puede ser un seudónimo que identifique la tabla. No puede ser el nombre de una tabla de catálogo (SQLSTATE 42832), una tabla de resumen (SQLSTATE 42997), una tabla con tipo (SQLSTATE 42997), un apodo ni un objeto distinto de una tabla o seudónimo (SQLSTATE 42809).

##### *identificador-tabla-nuevo*

Especifica el nuevo nombre para la tabla sin un nombre de esquema. Se utiliza el nombre de esquema del *nombre-tabla* para calificar el nuevo nombre de la tabla. El nombre calificado *no* debe identificar una tabla, una vista ni un seudónimo que ya exista en la base de datos (SQLSTATE 42710).

#### Normas

La tabla fuente debe cumplir estas condiciones:

- No estar referenciada en ninguna definición de vista existente o definición de tabla de resumen
- No estar referenciada en ninguna sentencia de SQL activada de desencadenantes existentes ni ser la tabla sujeto de un desencadenante existente
- No estar referenciada en una función SQL
- No tener ninguna restricción de comprobación



- No tener ninguna columna generada distinta de la columna de identidad
- No ser una tabla padre ni una tabla dependiente en ninguna restricción de integridad de referencia
- No ser el ámbito de ninguna columna de referencia existente.

Se devuelve un error (SQLSTATE 42986) si la tabla fuente viola una o más de esas condiciones.

### Notas

- Las entradas del catálogo se actualizan para reflejar el nuevo nombre de tabla.
- *Todas las* autorizaciones asociadas al nombre de tabla fuente se *transfieren* al nuevo nombre de tabla (las tablas del catálogo de autorizaciones se actualizan adecuadamente).
- Los índices definidos en la tabla fuente se *transfieren* a la nueva tabla (las tablas de catálogo de índice se actualizan del modo apropiado).
- Cualquier paquete que sea dependiente de la tabla fuente se invalida.
- Si para el *nombre-tabla* se utiliza un seudónimo, éste debe resolverse en un nombre de tabla. La tabla se redenomina dentro del esquema de la tabla. El seudónimo no se cambia por la sentencia RENAME y continúa haciendo referencia al anterior nombre de tabla.
- Una tabla con una clave primaria o restricciones de unicidad debe redeterminarse si ninguna clave foránea hace referencia a la clave primaria ni a las restricciones de unicidad.

### Ejemplo

Cambie el nombre de la tabla EMP por EMPLOYEE.

```
RENAME TABLE EMP TO EMPLOYEE
```

## RENAME TABLESPACE

---

### RENAME TABLESPACE

La sentencia RENAME TABLESPACE cambia el nombre de un espacio de tablas existente.

#### Invocación

Esta sentencia se puede intercalar en un programa de aplicación o se puede emitir utilizando sentencias de SQL dinámicas. Es una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

#### Autorización

El ID de autorización de la sentencia debe tener autorización SYSADM o SYSCTRL.

#### Sintaxis

►►—RENAME—TABLESPACE—*nombre-espacio-tablas-fuente*—T0—*nombre-espacio-tablas-destino*—►►

#### Descripción

*nombre-espacio-tablas-fuente*

Especifica, en forma de nombre que consta de un solo elemento, el espacio de tablas existente que se debe red denominar. Es un identificador de SQL (ordinario o delimitado). El nombre debe identificar un espacio de tablas que ya exista en el catálogo (SQLSTATE 42704).

*nombre-espacio-tablas-destino*

Especifica el nuevo nombre del espacio de tablas, en forma de nombre formado por un solo elemento. Es un identificador de SQL (ordinario o delimitado). El nombre *no* debe identificar un espacio de tablas que ya exista en el catálogo (SQLSTATE 42710) y no puede comenzar con 'SYS' (SQLSTATE 42939).

#### Normas

- El espacio de tablas SYSCATSPACE no se puede red denominar (SQLSTATE 42832).
- Los espacios de tablas cuyo estado sea "recuperación pendiente" o "recuperación en proceso" no se pueden red denominar (SQLSTATE 55039)

#### Notas

- Cuando se red denomina un espacio de tablas, se actualiza su tiempo mínimo de recuperación hasta el momento en que se cambió el nombre. Esto implica que una recuperación hecha a nivel de espacio de tablas debe hacerse como mínimo hasta alcanzar ese momento.

- El nuevo nombre del espacio de tablas se debe utilizar cuando se restaura un espacio de tablas a partir de una imagen de copia de seguridad, cuando el cambio de nombre se hizo después de crear la copia de seguridad. Consulte los manuales *Administrative API Reference* o *Consulta de mandatos* para obtener más información sobre la restauración de copias de seguridad.

### Ejemplo

Cambie el nombre del espacio de tablas USERSPACE1 a DATA2000:

```
RENAME TABLESPACE USERSPACE1 TO DATA2000
```

## REVOKE (autorizaciones de bases de datos)

### REVOKE (autorizaciones de bases de datos)

Esta forma de la sentencia REVOKE revoca las autorizaciones que se aplican a toda la base de datos.

#### Invocación

Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que se puede preparar dinámicamente. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

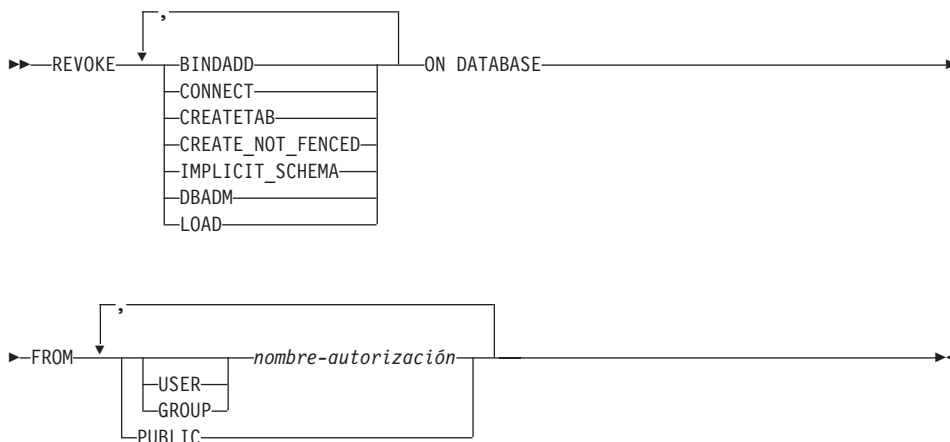
#### Autorización

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización DBADM
- Autorización SYSADM

Para revocar la autorización DBADM, se necesita la autorización SYSADM.

#### Sintaxis



#### Descripción

##### BINDADD

Revoca la autorización para crear paquetes. El creador de un paquete tiene automáticamente el privilegio CONTROL en dicho paquete y conserva este privilegio incluso si se revoca posteriormente la autorización BINDADD.

## REVOKE (autorizaciones de bases de datos)

La autorización BINDADD no puede revocarse desde un *nombre-autorización* que posea la autorización DBADM sin revocar también la autorización DBADM.

### CONNECT

Revoca la autorización para acceder a la base de datos.

La revocación de la autorización CONNECT de un usuario no afecta a ningún privilegio que se haya otorgado a dicho usuario en objetos de la base de datos. Si con posterioridad se vuelve a otorgar al usuario la autorización CONNECT, todos los privilegios que tenía anteriormente siguen siendo válidos (suponiendo que no se hayan revocado explícitamente).

La autorización CONNECT no puede revocarse de un *nombre-autorización* que contenga la autorización DBADM sin revocar también la autorización DBADM (SQLSTATE 42504).

### CREATETAB

Revoca la autorización para crear tablas. El creador de una tabla tiene automáticamente el privilegio CONTROL en dicha tabla y conserva este privilegio incluso si se revoca la autorización CREATETAB con posterioridad.

La autorización CREATETAB no se puede revocar de un *nombre-autorización* que tiene la autorización DBADM sin revocar también la autorización DBADM (SQLSTATE 42504).

### CREATE\_NOT\_FENCED

Revoca la autorización para registrar funciones que se ejecutan en el proceso del gestor de bases de datos. Sin embargo, una vez que se haya registrado la función como no limitada, continúa ejecutándose de esta manera incluso si CREATE\_NOT\_FENCED se revoca con posterioridad del ID de autorización que ha registrado la función.

La autorización CREATE\_NOT\_FENCED no se puede revocar de un *nombre-autorización* que tiene la autorización DBADM sin también revocar la autorización DBADM (SQLSTATE 42504).

### IMPLICIT\_SCHEMA

Revoca la autorización para crear implícitamente un esquema. No afecta la posibilidad de crear objetos en los esquemas existentes o de procesar una sentencia CREATE SCHEMA.

### DBADM

Revoca la autorización DBADM.

La autorización DBADM no puede revocarse de PUBLIC (porque no puede otorgarse a PUBLIC).

## REVOKE (autorizaciones de bases de datos)

La revocación de la autorización DBADM no revoca automáticamente ningún privilegio que tuviera el nombre-autorización en los objetos de la base de datos, ni revoca la autorización BINDADD, CONNECT, CREATETAB, IMPLICIT\_SCHEMA, LOAD o CREATE\_NOT\_FENCED.

### LOAD

Revoca la autorización para cargar en la base de datos.

### FROM

Indica a quién se revocan las autorizaciones.

### USER

Especifica que el *nombre-autorización* identifica a un usuario.

### GROUP

Especifica que el *nombre-autorización* identifica un nombre de grupo.

*nombre-autorización*,...

Lista uno o varios ID de autorización.

El ID de autorización de la propia sentencia REVOKE no se puede utilizar (SQLSTATE 42502). No es posible revocar las autorizaciones de un *nombre-autorización* que sea igual que el ID de autorización de la sentencia REVOKE.

### PUBLIC

Revoca las autorizaciones de PUBLIC.

## Normas

- Si no se especifica USER ni GROUP, entonces:
  - Si todas las filas para el destinatario de la operación de otorgar en la vista de catálogo SYSCAT.DBAUTH tienen GRANTEETYPE de U, se supone USER.
  - Si todas las filas tienen GRANTEETYPE de G, entonces se supone GROUP.
  - Si algunas filas tienen U y otras tienen G, se genera un error (SQLSTATE 56092).
  - Si se utiliza la autenticación DCE, se genera un error (SQLSTATE 56092).

## Notas

- La revocación de un privilegio específico no revoca necesariamente la posibilidad de realizar la acción. Un usuario puede seguir con su tarea si PUBLIC o un grupo tienen otros privilegios o si tienen una autorización de nivel superior como, por ejemplo, DBADM.

## Ejemplos

*Ejemplo 1:* Dado que USER6 sólo es un usuario y no un grupo, revoque el privilegio para crear las tablas del usuario USER6.

```
REVOKE CREATETAB ON DATABASE FROM USER6
```

## REVOKE (autorizaciones de bases de datos)

*Ejemplo 2:* Revoque la autorización BINDADD en la base de datos de un grupo denominado D024. Hay dos filas en la vista de catálogo SYSCAT.DBAUTH para quien se otorga; una con GRANTEETYPE de U y otra con GRANTEETYPE de G.

```
REVOKE BINDADD ON DATABASE FROM GROUP D024
```

En este caso, debe especificarse la palabra clave GROUP; de lo contrario, se producirá un error (SQLSTATE 56092).

## REVOKE (privilegios de índice)

---

### REVOKE (privilegios de índice)

Esta forma de la sentencia REVOKE revoca el privilegio CONTROL en un índice.

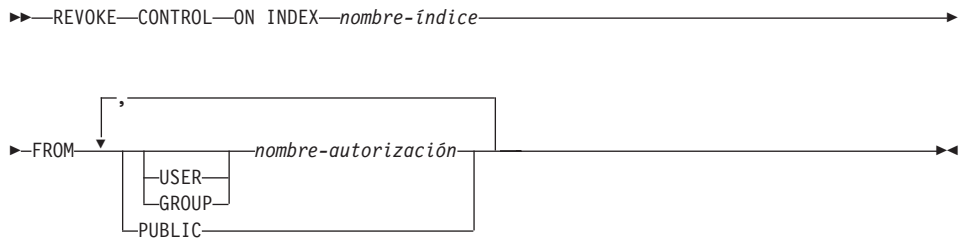
#### Invocación

Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

#### Autorización

El ID de autorización de la sentencia debe tener la autorización SYSADM o DBADM (SQLSTATE 42501).

#### Sintaxis



#### Descripción

##### CONTROL

Revoca el privilegio para eliminar el índice. Este es el privilegio CONTROL para índices, que se otorga automáticamente a los creadores de índices.

##### ON INDEX *nombre-índice*

Especifica el nombre del índice en el que se ha de revocar el privilegio CONTROL.

##### FROM

Indica de quién se han de revocar los privilegios.

##### USER

Especifica que el *nombre-autorización* identifica a un usuario.

##### GROUP

Especifica que el *nombre-autorización* identifica un nombre de grupo.

*nombre-autorización*,...

Lista uno o varios ID de autorización.



## REVOKE (privilegios de índice)

El ID de autorización de la propia sentencia REVOKE no se puede utilizar (SQLSTATE 42502). No se pueden revocar privilegios para un *nombre-autorización* que sea igual que el ID de autorización de la sentencia REVOKE.

### PUBLIC

Revoca los privilegios de PUBLIC.

### Normas

- Si no se especifica USER ni GROUP, entonces:
  - Si todas las filas para el destinatario de la operación de otorgar en la vista de catálogo SYSCAT.INDEXAUTH tienen GRANTEETYPE de U, entonces se supone USER.
  - Si todas las filas tienen GRANTEETYPE de G, entonces se supone GROUP.
  - Si algunas filas tienen U y otras tienen G, se genera un error (SQLSTATE 56092).
  - Si se utiliza la autenticación DCE, se genera un error (SQLSTATE 56092).

### Notas

- La revocación de un privilegio específico no revoca necesariamente la posibilidad de realizar la acción. Un usuario puede seguir con su tarea si PUBLIC o un grupo tienen otros privilegios o si tienen autorizaciones como ALTERIN en el esquema de un índice.

### Ejemplos

*Ejemplo 1:* Dado que USER4 es sólo un usuario y no un grupo, revoque el privilegio para eliminar un índice DEPTIDX del usuario USER4.

```
REVOKE CONTROL ON INDEX DEPTIDX FROM USER4
```

*Ejemplo 2:* Revoque el privilegio para eliminar un índice LUNCHITEMS del usuario CHEF y del grupo WAITERS.

```
REVOKE CONTROL ON INDEX LUNCHITEMS
FROM USER CHEF, GROUP WAITERS
```

## REVOKE (privilegios de paquete)

### REVOKE (privilegios de paquete)

Esta forma de la sentencia REVOKE revoca los privilegios CONTROL, BIND y EXECUTE en un paquete.

#### Invocación

Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

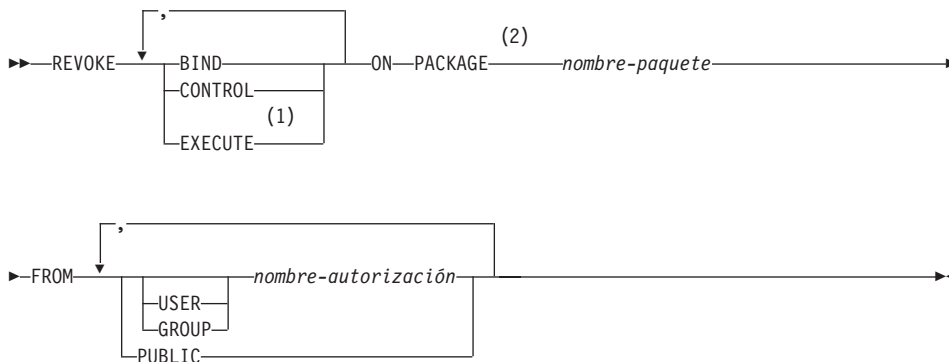
#### Autorización

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Privilegio CONTROL para el paquete referenciado
- Autorización SYSADM o DBADM.

Para revocar el privilegio CONTROL, es necesaria la autorización SYSADM o DBADM.

#### Sintaxis



#### Notas:

- 1 RUN se puede utilizar como sinónimo de EXECUTE.
- 2 PROGRAM puede utilizarse como sinónimo de PACKAGE.

#### Descripción

##### BIND

Revoca el privilegio para ejecutar BIND o REBIND en el paquete de referencia.

## REVOKE (privilegios de paquete)

Los privilegios BIND no pueden revocarse de un *nombre-autorización* que tenga el privilegio CONTROL en el paquete sin revocar también el privilegio CONTROL.

### CONTROL

Revoca el privilegio para eliminar el paquete y para extender los privilegios de paquete a otros usuarios.

La revocación de CONTROL no revoca los demás privilegios del paquete.

### EXECUTE

Revoca el privilegio para ejecutar el paquete.

El privilegio EXECUTE no puede revocarse de un *nombre-autorización* que tiene el privilegio CONTROL en el paquete sin revocar también el privilegio CONTROL.

### ON PACKAGE *nombre-paquete*

Especifica el paquete en el que se revocan los privilegios.

### FROM

Indica de quién se han de revocar los privilegios.

### USER

Especifica que el *nombre-autorización* identifica a un usuario.

### GROUP

Especifica que el *nombre-autorización* identifica un nombre de grupo.

*nombre-autorización*,...

Lista uno o varios ID de autorización.

El ID de autorización de la propia sentencia REVOKE no se puede utilizar (SQLSTATE 42502). No se pueden revocar privilegios para un *nombre-autorización* que sea igual que el ID de autorización de la sentencia REVOKE.

### PUBLIC

Revoca los privilegios de PUBLIC.

## Normas

- Si no se especifica USER ni GROUP, entonces:
  - Si todas las filas para el destinatario de la operación de otorgar en la vista de catálogo SYSCAT.PACKAGEAUTH tienen GRANTEETYPE de U, entonces se supone USER.
  - Si todas las filas tienen GRANTEETYPE de G, entonces se supone GROUP.
  - Si algunas filas tienen U y otras tienen G, se genera un error (SQLSTATE 56092).
  - Si se utiliza la autenticación DCE, se genera un error (SQLSTATE 56092).

## REVOKE (privilegios de paquete)

### Notas

- La revocación de un privilegio específico no revoca necesariamente la posibilidad de realizar la acción. Un usuario puede seguir con su tarea si PUBLIC o un grupo poseen otros privilegios, o si tienen privilegios como, por ejemplo, ALTERIN en el esquema de un paquete.

### Ejemplos

*Ejemplo 1:* Revoque el privilegio EXECUTE en el paquete CORPDATA.PKGA de PUBLIC.

```
REVOKE EXECUTE
ON PACKAGE CORPDATA.PKGA
FROM PUBLIC
```

*Ejemplo 2:* Revoque la autorización CONTROL en el paquete RRSP\_PKG para el usuario FRANK y para PUBLIC.

```
REVOKE CONTROL
ON PACKAGE RRSP_PKG
FROM USER FRANK, PUBLIC
```

## REVOKE (privilegios de esquema)

Esta forma de la sentencia REVOKE revoca los privilegios en un esquema.

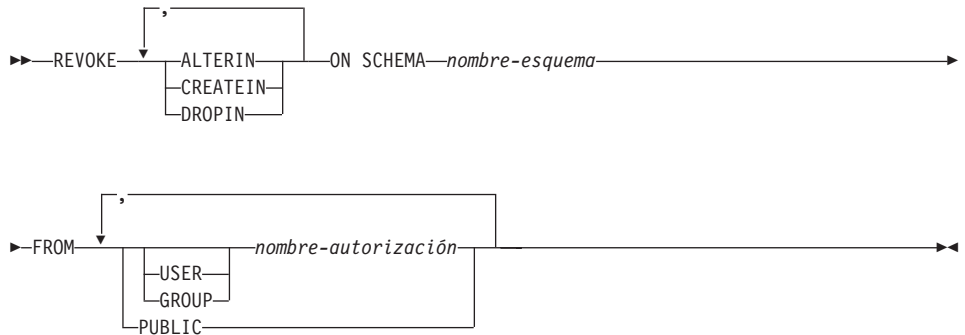
### Invocación

Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

### Autorización

El ID de autorización de la sentencia debe tener la autorización SYSADM o DBADM (SQLSTATE 42501).

### Sintaxis



### Descripción

#### ALTERIN

Revoca el privilegio para modificar o comentar los objetos del esquema.

#### CREATEIN

Revoca el privilegio para crear objetos en el esquema.

#### DROPIN

Revoca el privilegio para eliminar objetos en el esquema.

#### ON SCHEMA *nombre-esquema*

Especifica el nombre del esquema en el que se han de revocar los privilegios.

#### FROM

Indica de quién se han de revocar los privilegios.

#### USER

Especifica que el *nombre-autorización* identifica a un usuario.

## REVOKE (privilegios de esquema)

### GROUP

Especifica que el *nombre-autorización* identifica un nombre de grupo.

*nombre-autorización*,...

Lista uno o varios ID de autorización.

El ID de autorización de la propia sentencia REVOKE no se puede utilizar (SQLSTATE 42502). No se pueden revocar privilegios para un *nombre-autorización* que sea igual que el ID de autorización de la sentencia REVOKE.

### PUBLIC

Revoca los privilegios de PUBLIC.

## Normas

- Si no se especifica USER ni GROUP, entonces:
  - Si todas las filas del destinatario de la operación de otorgar en la vista de catálogo SYSCAT.SCHEMAAUTH tienen GRANTEETYPE de U, se supone USER.
  - Si todas las filas tienen GRANTEETYPE de G, entonces se supone GROUP.
  - Si algunas filas tienen U y otras tienen G, se genera un error (SQLSTATE 56092).
  - Si se utiliza la autenticación DCE, se genera un error (SQLSTATE 56092).

## Notas

- La revocación de un privilegio específico no revoca necesariamente la posibilidad de realizar la acción. Un usuario puede seguir con su tarea si PUBLIC o un grupo tienen otros privilegios o si tienen una autorización de nivel superior como, por ejemplo, DBADM.

## Ejemplos

*Ejemplo 1:* Suponiendo que USER4 sea sólo un usuario y no un grupo, revoque el privilegio para crear objetos en el esquema DEPTIDX del usuario USER4.

```
REVOKE CREATEIN ON SCHEMA DEPTIDX FROM USER4
```

*Ejemplo 2:* Revoque el privilegio de eliminar objetos en el esquema LUNCH del usuario CHEF y del grupo WAITERS.

```
REVOKE DROPIN ON SCHEMA LUNCH
FROM USER CHEF, GROUP WAITERS
```

## REVOKE (Privilegios de servidor)

Este formato de la sentencia REVOKE revoca el privilegio para acceder y utilizar una fuente de datos especificada en la modalidad de paso a través.

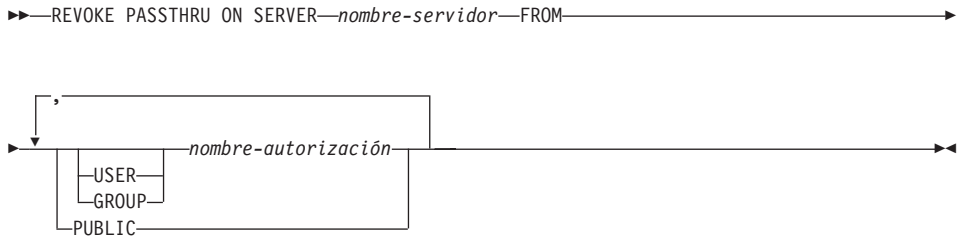
### Invocación

Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que se puede preparar dinámicamente. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

### Autorización

El ID de autorización de la sentencia debe tener la autorización SYSADM o DBADM.

### Sintaxis



### Descripción

**SERVER** *nombre-servidor*

Nombra la fuente de datos para la cual se está revocando el privilegio para utilizarse en modalidad de paso a través. *nombre-servidor* debe identificar una fuente de datos que esté descrita en el catálogo.

**FROM**

Especifica a quién se revoca el privilegio.

**USER**

Especifica que el *nombre-autorización* identifica a un usuario.

**GROUP**

Especifica que el *nombre-autorización* identifica un nombre de grupo.

*nombre-autorización*,...

Lista los ID de autorización de uno o varios usuarios o grupos.

El ID de autorización de la propia sentencia REVOKE no se puede utilizar (SQLSTATE 42502). No se pueden revocar privilegios para un *nombre-autorización* que sea igual que el ID de autorización de la sentencia REVOKE.

## REVOKE (Privilegios de servidor)

### PUBLIC

Revoca a todos los usuarios el privilegio de realizar un paso a través para *nombre-servidor*.

### Ejemplos

*Ejemplo 1:* Revoque el privilegio que USER6 tiene para utilizar la modalidad de paso a través para la fuente de datos MOUNTAIN.

```
REVOKE PASSTHRU ON SERVER MOUNTAIN FROM USER USER6
```

*Ejemplo 2:* Revoque el privilegio que el grupo D024 tiene para usar la modalidad de paso a través para la fuente de datos EASTWING.

```
REVOKE PASSTHRU ON SERVER EASTWING FROM GROUP D024
```

Los miembros del grupo D024 ya no podrán utilizar su ID de grupo para realizar un paso a través para EASTWING. Pero si cualquier miembro tiene el privilegio para usar el paso a través para EASTWING con su propio ID de usuario, conservará este privilegio.



## REVOKE (privilegios de apodo, vista o tabla)

Esta forma de la sentencia REVOKE revoca privilegios en una tabla, vista o apodo.

### Invocación

Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

### Autorización

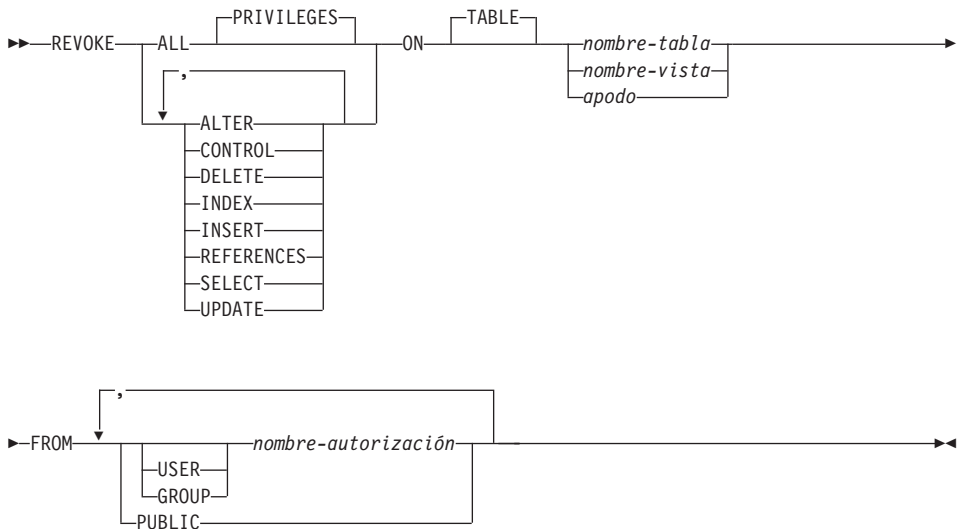
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Privilegio CONTROL para la tabla, vista o apodo referenciado.

Para revocar el privilegio CONTROL, es necesaria la autorización SYSADM o DBADM.

Para revocar los privilegios sobre tablas y vistas de catálogo es necesaria la autorización SYSADM o DBADM.

### Sintaxis



## REVOKE (privilegios de apodo, vista o tabla)

### Descripción

#### ALL o ALL PRIVILEGES

Revoca todos los privilegios que tiene un nombre-autorización para las tablas, vistas o apodos especificados.

Si no se utiliza ALL, deben utilizarse una o varias de las palabras clave listadas abajo. Cada palabra clave revoca el privilegio descrito, pero sólo si se aplica a las tablas o vistas nombradas en la cláusula ON. No se debe especificar la misma palabra clave más de una vez.

#### ALTER

Revoca el privilegio para añadir columnas a la definición de la tabla base; crear o eliminar una clave primaria o restricción de unicidad en la tabla; crear o eliminar una clave foránea en la tabla; añadir/cambiar un comentario en la tabla, vista o apodo; crear o eliminar una restricción de comprobación; crear un desencadenante; añadir, restablecer o eliminar una opción de columna para un apodo; o cambiar nombres de columna de apodo o tipos de datos.

#### CONTROL

Revoca la capacidad para eliminar la tabla base, vista o apodo y para ejecutar el programa de utilidad RUNSTATS sobre la tabla y los índices.

La revocación del privilegio CONTROL en un *nombre-autorización* no revoca otros privilegios otorgados al usuario de dicho objeto.

#### DELETE

Revoca el privilegio para suprimir filas de la tabla o de la vista actualizable.

#### INDEX

Revoca el privilegio para crear un índice en la tabla o una especificación de índice en el apodo. El creador de un índice o de una especificación de índice tiene automáticamente el privilegio CONTROL en el índice o en la especificación de índice (que autoriza al creador a eliminar el índice o la especificación de índice). Así mismo, el creador mantiene este privilegio incluso si se revoca el privilegio INDEX.

#### INSERT

Revoca el privilegio para insertar filas en la tabla o vista actualizable y el privilegio para ejecutar el programa de utilidad IMPORT.

#### REFERENCES

Revoca el privilegio para crear o eliminar una clave foránea que haga referencia a la tabla como padre. Cualquier privilegio REFERENCES de nivel de columna también se revoca.

## REVOKE (privilegios de apodo, vista o tabla)

### SELECT

Revoca el privilegio para recuperar filas de la tabla o vista, para crear una vista en una tabla y para ejecutar el programa de utilidad EXPORT sobre la tabla o vista.

La revocación del privilegio SELECT puede provocar que algunas vistas se marquen como no operativas. Para obtener información sobre vistas no operativas, consulte el apartado “Notas” en la página 966.

### UPDATE

Revoca el privilegio para actualizar filas de la tabla o vista actualizable. Cualquier privilegio UPDATE de nivel de columna también se revoca.

### ON TABLE *nombre-tabla* o *nombre-vista* o *apodo*

Especifica la tabla, vista o apodo en los que se han de revocar los privilegios. El *nombre-tabla* no puede ser una tabla temporal declarada (SQLSTATE 42995).

### FROM

Indica de quién se han de revocar los privilegios.

### USER

Especifica que el *nombre-autorización* identifica a un usuario.

### GROUP

Especifica que el *nombre-autorización* identifica un nombre de grupo.

*nombre-autorización*,...

Lista uno o varios ID de autorización.

El ID de la propia sentencia REVOKE no se puede utilizar (SQLSTATE 42502). No se pueden revocar privilegios para un *nombre-autorización* que sea igual que el ID de autorización de la sentencia REVOKE.

### PUBLIC

Revoca los privilegios de PUBLIC.

## Normas

- Si no se especifica USER ni GROUP, entonces:
  - Si todas las filas para el destinatario en la vista de catálogo SYSCAT.TABAUTH y SYSCAT.COLAUTH tienen GRANTEETYPE de U, entonces se supone USER.
  - Si todas las filas tienen GRANTEETYPE de G, entonces se supone GROUP.
  - Si algunas filas tienen U y otras tienen G, se genera un error (SQLSTATE 56092).
  - Si se utiliza la autenticación DCE, se genera un error (SQLSTATE 56092).

## REVOKE (privilegios de apodo, vista o tabla)

### Notas

- Si se revoca un privilegio del *nombre-autorización* utilizado para crear una vista (se denomina DEFINER de la vista en SYSCAT.VIEWS), dicho privilegio también se revoca de las vistas dependientes.
- Si el DEFINER de la vista pierde un privilegio SELECT sobre algún objeto del que dependa la definición de vista (o se elimina un objeto del que dependa la definición de vista (o se vuelve no operativo en el caso de otra vista)), la vista se volverá no operativa (consulte el apartado “Notas” de “CREATE VIEW” en la página 957 para obtener información sobre las vistas no operativas).

Sin embargo, si DBADM o SYSADM revoca explícitamente todos los privilegios del DEFINER sobre la vista, el registro del DEFINER no aparecerá en SYSCAT.TABAUTH, pero no le sucederá nada a la vista - continuará operativa.

- Los privilegios en vistas no operativas no pueden revocarse.
- Todos los paquetes que dependen de un objeto para el que se revoca un privilegio se marcan como no válidos. Un paquete continúa siendo no válido hasta que se ejecuta satisfactoriamente una operación de enlace lógico o de volver a enlazar lógicamente en la aplicación, o la aplicación se ejecuta y el gestor de bases de datos vuelve a enlazar la aplicación satisfactoriamente (utilizando la información almacenada en los catálogos). Los paquetes marcados como no válidos debido a una revocación pueden volverse a enlazar satisfactoriamente sin ninguna operación de otorgar adicional.

Por ejemplo, si un paquete propiedad de USER1 contiene SELECT de la tabla T1 y se revoca el privilegio SELECT para la tabla T1 del USER1, el paquete se marcará como no válido. Si se vuelve a otorgar la autorización SELECT, o si el usuario tiene la autorización DBADM, el paquete se vuelve a enlazar satisfactoriamente cuando se ejecuta.

- Paquetes, desencadenantes o vistas que incluyen la utilización de OUTER(Z) en la cláusula FROM, dependen de tener el privilegio SELECT en cada subtabla o subvista de Z. De modo similar, paquetes, desencadenantes o vistas que incluyen la utilización de Deref(Y) donde Y es un tipo de referencia con una vista o tabla de destino Z, dependen de tener el privilegio SELECT en cada subtabla o subvista de Z. Si se revoca uno de estos privilegios SELECT, los paquetes se invalidan y los desencadenantes y vistas pasan a ser no operativas.
- Los privilegios de tabla, vista o apodo no pueden revocarse de un *nombre-autorización* con CONTROL en el objeto sin revocar también el privilegio CONTROL (SQLSTATE 42504).
- La revocación de un privilegio específico no revoca necesariamente la posibilidad de realizar la acción. Un usuario puede seguir con su tarea si PUBLIC o un grupo poseen otros privilegios, o si tienen privilegios como, por ejemplo, ALTERIN en el esquema de una tabla o vista.

## REVOKE (privilegios de apodo, vista o tabla)

- Si el DEFINER de la tabla de resumen pierde un privilegio SELECT sobre una tabla de la que dependa la definición de tabla de resumen (o se elimina una tabla de la que dependa la definición de tabla de resumen), la tabla de resumen se volverá no operativa (consulte las “Notas” en la página 881 para obtener información sobre las tablas de resumen no operativas).

Sin embargo, si DBADM o SYSADM revoca explícitamente todos los privilegios del DEFINER sobre la tabla de resumen, el registro de SYSTABAUTH para el DEFINER se suprimirá, pero no le sucederá nada a la tabla de resumen - continuará operativa.

- Revocar privilegios de apodo no tiene efecto sobre los privilegios de objeto de la fuente de datos (tabla o vista).
- La revocación del privilegio SELECT para una tabla o vista que esté referenciada *directamente* o *indirectamente* en una función de SQL puede no ser efectiva si la función no se puede eliminar debido a que algún otro objeto depende de la función (SQLSTATE 42893).

**Nota:** El apartado “Normas” en la página 1023 lista las dependencias que dichos objetos como, por ejemplo, las tablas y vistas, pueden tener entre sí.

### Ejemplos

*Ejemplo 1:* Revoque el privilegio SELECT en la tabla EMPLOYEE del usuario ENGLS. Hay una fila en la vista de catálogo SYSCAT.TABAUTH para esta tabla y el destinatario de la operación de otorgar y el valor de GRANTEETYPE es U.

```
REVOKE SELECT
ON TABLE EMPLOYEE
FROM ENGLS
```

*Ejemplo 2:* Revoque los privilegios de actualización en la tabla EMPLOYEE que se han otorgado previamente a todos los usuarios locales. Tenga en cuenta que no afecta a los usuarios a los que se les ha otorgado de manera específica.

```
REVOKE UPDATE
ON EMPLOYEE
FROM PUBLIC
```

*Ejemplo 3:* Revoque todos los privilegios en la tabla EMPLOYEE de los usuarios PELLOW y MLI y del grupo PLANNERS.

```
REVOKE ALL
ON EMPLOYEE
FROM USER PELLOW, USER MLI, GROUP PLANNERS
```

## REVOKE (privilegios de apodo, vista o tabla)

*Ejemplo 4:* Revoque el privilegio SELECT en la tabla CORPDATA.EMPLOYEE de un usuario llamado JOHN. Hay una fila en la vista de catálogo SYSCAT.TABAUTH para esta tabla y el destinatario de la operación de otorgar y el valor de GRANTEETYPE es U.

```
REVOKE SELECT
ON CORPDATA.EMPLOYEE FROM JOHN
```

o

```
REVOKE SELECT
ON CORPDATA.EMPLOYEE FROM USER JOHN
```

Tenga en cuenta que el intento de revocar el privilegio de GROUP JOHN daría como resultado un error, ya que el privilegio no estaba otorgado previamente a GROUP JOHN.

*Ejemplo 5:* Revoque el privilegio SELECT en la tabla CORPDATA.EMPLOYEE de un grupo llamado JOHN. Hay una fila en la vista de catálogo SYSCAT.TABAUTH para esta tabla y el destinatario de la operación de otorgar y el valor GRANTEETYPE es G.

```
REVOKE SELECT
ON CORPDATA.EMPLOYEE FROM JOHN
```

o

```
REVOKE SELECT
ON CORPDATA.EMPLOYEE FROM GROUP JOHN
```

*Ejemplo 6:* Revoque el privilegio SHAWN de usuario para crear una especificación de índice en el apodo ORAREM1.

```
REVOKE INDEX
ON ORAREM1 FROM USER SHAWN
```

## REVOKE (privilegios para espacios de tablas)

Esta forma de la sentencia REVOKE revoca el privilegio USE para una tabla.

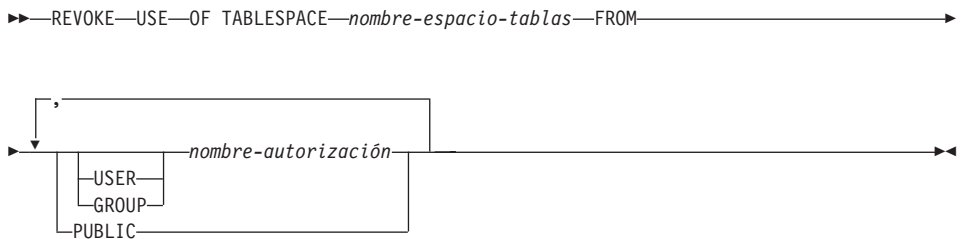
### Invocación

Esta sentencia se puede intercalar en un programa de aplicación o se puede emitir utilizando sentencias de SQL dinámicas. Es una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si se aplica la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

### Autorización

El ID de autorización de la sentencia debe tener la autorización SYSADM, SYSCTRL o DBADM (SQLSTATE 42501).

### Sintaxis



### Descripción

#### USE

Revoca el privilegio para especificar, de forma explícita o por omisión, el espacio de tablas al crear una tabla.

#### OF TABLESPACE *nombre-espacio-tablas*

Identifica el espacio de tablas para el que debe revocar el privilegio USE. El espacio de tablas no puede ser SYSCATSPACE (SQLSTATE 42838) ni un espacio de tablas temporal del sistema (SQLSTATE 42809).

#### FROM

Especifica a quién se revoca el privilegio USE.

#### USER

Especifica que el *nombre-autorización* identifica a un usuario.

#### GROUP

Especifica que el *nombre-autorización* identifica un nombre de grupo.

#### *nombre-autorización*

Lista uno o varios ID de autorización.

## REVOKE (privilegios para espacios de tablas)

El ID de autorización de la propia sentencia REVOKE no se puede utilizar (SQLSTATE 42502). No es posible revocar los privilegios de un *nombre-autorización* que es igual que el ID de autorización de la sentencia REVOKE.

### **PUBLIC**

Revoca el privilegio USE a PUBLIC.

### **Normas**

- Si no se especifica USER ni GROUP, entonces:
  - Si todas las filas de la vista de catálogo SYSCAT.TBSPACEAUTH correspondientes al usuario autorizado tienen U como GRANTEETYPE, se supone USER.
  - Si todas las filas tienen G como GRANTEETYPE, entonces se supone GROUP.
  - Si algunas filas tienen U y otras tienen G, se produce un error (SQLSTATE 56092).
  - Si se utiliza la autenticación DCE, se produce un error (SQLSTATE 56092).

### **Notas**

- La revocación del privilegio USE no revoca necesariamente la capacidad para crear tablas en ese espacio de tablas. Un usuario puede todavía crear tablas en ese espacio de tablas si el privilegio USE se otorga a PUBLIC o a un grupo, o si el usuario tiene una autorización de nivel superior, tal como DBADM.

### **Ejemplos**

*Ejemplo 1:* Revocación del privilegio del usuario BOBBY para crear tablas en el espacio de tablas PLANS.

```
REVOKE USE OF TABLESPACE PLANS FROM USER BOBBY
```



## ROLLBACK

La sentencia ROLLBACK se utiliza para restituir los cambios que se han hecho en la base de datos dentro de una unidad de trabajo o punto de salvar.

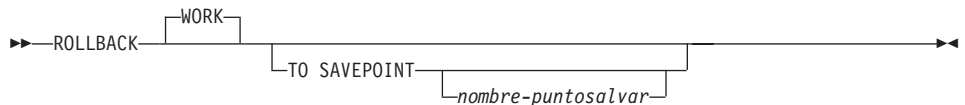
### Invocación

Esta sentencia puede incluirse en un programa de aplicación o bien emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que se puede preparar dinámicamente.

### Autorización

No se necesita.

### Sintaxis



### Descripción

La unidad de trabajo en la que se ejecuta la sentencia ROLLBACK se termina y se inicia una nueva unidad de trabajo. Se restituyen todos los cambios realizados en la base de datos durante la unidad de trabajo.

Sin embargo, las sentencias siguientes no están bajo el control de transacción y los cambios que realicen son independientes de la emisión de la sentencia ROLLBACK:

- SET CONNECTION
- SET CURRENT DEGREE
- SET CURRENT DEFAULT TRANSFORM GROUP
- SET CURRENT EXPLAIN MODE
- SET CURRENT EXPLAIN SNAPSHOT
- SET CURRENT PACKAGESET
- SET CURRENT QUERY OPTIMIZATION
- SET CURRENT REFRESH AGE
- SET EVENT MONITOR STATE
- SET PASSTHRU
- SET PATH
- SET SCHEMA
- SET SERVER OPTION

### TO SAVEPOINT

Indica que debe realizarse una retrotracción parcial (ROLLBACK TO SAVEPOINT). Si no existe ningún punto de salvar activo, se emite un

## ROLLBACK

error de SQL (SQLSTATE 3B502). Después de una sentencia ROLLBACK satisfactoria, el punto de salvar continúa existiendo. Si no se proporciona un *nombre-puntosalvar*, la retrotracción se realiza hasta el punto de salvar definido más recientemente.

Si se omite esta cláusula, la sentencia ROLLBACK WORK retrotrae la transacción completa. Adem, se liberan todos los puntos de salvar definidos dentro de la transacción.

### *nombre-puntosalvar*

Indica el punto de salvar hasta el que debe llegar la retrotracción. Después de una sentencia ROLLBACK satisfactoria, el punto de salvar definido por *nombre-puntosalvar* continúa existiendo. Si el nombre del punto de salvar no existe, se devuelve un error (SQLSTATE 3B001). Se deshacen los cambios que se han hecho en datos y esquemas desde que se definió el punto de salvar.

## Notas

- Cuando se ejecuta un ROLLBACK de la unidad de trabajo se liberan todos los bloqueos mantenidos. Se cierran todos los cursores abiertos. Se liberan todos los localizadores de LOB.
- La ejecución de la sentencia ROLLBACK no afecta a las sentencias SET que cambian los valores del registro especial ni a la sentencia RELEASE.
- Si el programa finaliza de forma anómala, la unidad de trabajo se retrotrae implícitamente.
- La colocación de sentencias en antememoria se ve afectada por la operación de retrotracción. Consulte el apartado “Notas” en la página 1037 para obtener más información.
- Los puntos de salvar no se pueden utilizar en contextos de ejecución atómica, tales como las sentencias compuestas atómicas y los desencadenantes atómicos.
- El efecto que una sentencia ROLLBACK TO SAVEPOINT tiene sobre los cursores depende de las sentencias contenidas en el punto de salvar.
  - Si el punto de salvar contiene un DDL del cual depende un cursor, el cursor se marca como no válido. Los intentos para utilizar ese cursor dan lugar a un error (SQLSTATE 57007).
  - En otro caso:
    - Si se hace referencia al cursor en el punto de salvar, el cursor permanece abierto y se coloca delante de la primera fila lógica de la tabla resultante.<sup>105</sup>
    - En otro caso, el cursor no queda afectado por ROLLBACK TO SAVEPOINT (permanece abierto y posicionado).

---

105. Se deberá realizar un FETCH antes de que se emita una sentencia UPDATE o DELETE de posición.

- Los nombres de sentencias preparadas dinámicamente siguen siendo válidos, aunque la sentencia puede volver a prepararse implícitamente, como resultado de operaciones DDL que se retrotraen dentro del punto de salvar.
- Las operaciones ROLLBACK TO SAVEPOINT eliminan cualquier tabla temporal declarada que aparezca mencionada dentro del punto de salvar. Si una tabla temporal declarada se modifica dentro del punto de salvar, se suprimen todas las filas de la tabla.
- Después de una sentencia ROLLBACK TO SAVEPOINT se conservan todos los bloqueos.
- Después de una operación ROLLBACK TO SAVEPOINT se conservan todos los localizadores de LOB.

### Ejemplo

Suprima las modificaciones realizadas desde el último punto de confirmación o retrotracción.

```
ROLLBACK WORK
```

## SAVEPOINT

---

### SAVEPOINT

Utilice la sentencia `SAVEPOINT` para definir un punto de salvar dentro de una transacción.

#### Invocación

Esta sentencia puede incluirse en un programa de aplicación (también en procedimientos almacenados) o emitirse de forma interactiva. Se trata de una sentencia ejecutable que puede prepararse dinámicamente.

#### Autorización

No se necesita.

#### Sintaxis

```
▶▶—SAVEPOINT—nombre-puntosalvar—┐
└─UNIQUE┘
▶—ON ROLLBACK RETAIN CURSORS—┐
└─ON ROLLBACK RETAIN LOCKS—┘▶▶
```

#### Descripción

*nombre-puntosalvar*

Nombre del *puntosalvar*.

#### UNIQUE

Esta opción indica que la aplicación no reutilizará este nombre de punto de salvar mientras el punto de salvar esté activo.

#### ON ROLLBACK RETAIN CURSORS

Especifica la respuesta del sistema al realizar una retrotracción hasta este punto de salvar, con respecto a las sentencias `OPEN CURSOR` procesadas después de la sentencia `SAVEPOINT`. La cláusula `RETAIN CURSORS` indica que, siempre que sea posible, la retrotracción hasta el punto de salvar no modificará los cursores. Para conocer los casos en los que la retrotracción hasta el punto de salvar afecta a los cursores, vea “`ROLLBACK`” en la página 1139.

#### ON ROLLBACK RETAIN LOCKS

Especifica la respuesta del sistema al realizar una retrotracción hasta este punto de salvar, con respecto a los bloqueos adquiridos después de definir el punto de salvar. No se hace un seguimiento de los bloqueos adquiridos desde que se definió el punto de salvar y no se retrotraen (liberan) al retrotraer hasta el punto de salvar.

## Normas

- Los puntos de salvar no se pueden anidar. Si se emite una sentencia SAVEPOINT y ya existe un punto de salvar definido, se produce un error (SQLSTATE 3B002).

## Notas

- La palabra clave UNIQUE está soportada para mantener la compatibilidad con DB2 Universal Database para OS/390. La información siguiente describe el comportamiento en DB2 Universal Database para OS/390.

Si un punto de salvar llamado *nombre-puntosalvar* ya existe dentro de la transacción, se devuelve un error (SQLSTATE 3B501). Mediante la omisión de la cláusula UNIQUE, la aplicación indica que el nombre del punto de salvar se puede volver a utilizar dentro de la transacción. Si *nombre-puntosalvar* ya existe en la transacción, se destruirá y se creará un punto de salvar nuevo llamado *nombre-puntosalvar*.

Suprimir un punto de salvar mediante la reutilización de su nombre para otro punto de salvar no es lo mismo que liberar el punto de salvar antiguo mediante la sentencia RELEASE SAVEPOINT. La reutilización de un nombre de punto de salvar produce la destrucción de sólo ese punto de salvar. La liberación de un punto de salvar mediante la sentencia RELEASE SAVEPOINT libera el punto de salvar especificado y todos los subsiguientes.

- Dentro de un punto de salvar, si un programa de utilidad, sentencia de SQL o mandato de DB2 ejecuta sentencias COMMIT intermitentes durante el proceso, el punto de salvar se libera implícitamente.
- La sentencia SET INTEGRITY de SQL tiene los mismos efectos que una sentencia DDL dentro de un punto de salvar.
- En una aplicación, las inserciones pueden estar en almacenamiento intermedio (es decir, la aplicación se precompiló utilizando la opción INSERT BUF). El almacenamiento intermedio se vacía cuando se emiten las sentencias SAVEPOINT, ROLLBACK o RELEASE TO SAVEPOINT.

## SELECT

---

### SELECT

La sentencia SELECT es una forma de consulta. Puede incluirse en un programa de aplicación o emitirse interactivamente. Para ver información detallada, consulte el apartado “sentencia-select” en la página 518 y “subselección” en la página 472.

## SELECT INTO

La sentencia `SELECT INTO` produce una tabla resultante que consta de como máximo una fila y asigna los valores de dicha fila a las variables del lenguaje principal. Si la tabla está vacía, la sentencia asigna +100 a `SQLCODE` y '02000' a `SQLSTATE` y no asigna valores a las variables del lenguaje principal. Si más de una fila satisface la condición de búsqueda, se termina el proceso de la sentencia y se produce un error (`SQLSTATE 21000`).

### Invocación

Esta sentencia sólo se puede incluir en un programa de aplicación. Se trata de una sentencia ejecutable que no puede prepararse de forma dinámica.

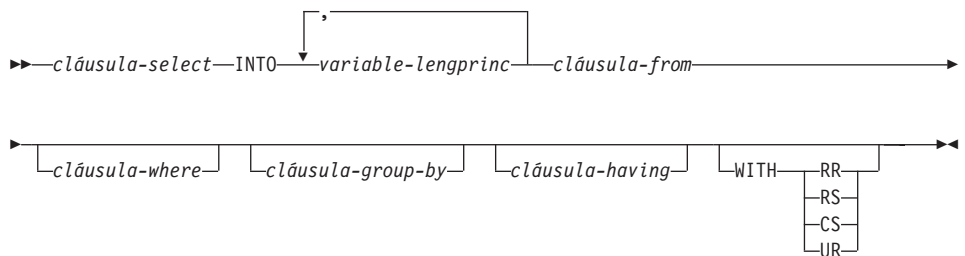
### Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir, para la tabla o vista a la que la sentencia `SELECT INTO` hace referencia, como mínimo uno de los siguientes:

- Privilegio `SELECT`
- Privilegio `CONTROL`
- Autorización `SYSADM` o `DBADM`

Los privilegios `GROUP` no se comprueban para las sentencias `SELECT INTO` estáticas.

### Sintaxis



### Descripción

Consulte el “Capítulo 5. Consultas” en la página 471 para obtener una descripción de la *cláusula-select*, *cláusula-from*, *cláusula-where*, *cláusula-group-by* y *cláusula-having*.

#### INTO

Introduce una lista de variables del lenguaje principal.

*variable-lengprinc*

Identifica una variable que está descrita en el programa bajo las reglas para la declaración de variables del lenguaje principal.

## SELECT INTO

El primer valor de la fila del resultado se asigna a la primera variable de la lista, el segundo valor a la segunda variable, etcétera. Si el número de variables del lenguaje principal es menor que el número de valores de columna, se asigna el valor 'W' al campo SQLWARN3 de la SQLCA. Consulte el “Apéndice B. Comunicaciones SQL (SQLCA)” en la página 1261.)

Cada asignación a una variable se realiza de acuerdo a las reglas descritas en el “Asignaciones y comparaciones” en la página 105. Las asignaciones se realizan en secuencia en la lista.

Si se produce un error, no se asigna ningún valor a la variable del lenguaje principal.

### WITH

Especifica el nivel de aislamiento en el que se ejecuta la sentencia SELECT INTO.

#### RR

Lectura repetible

#### RS

Estabilidad de lectura

#### CS

Estabilidad del cursor

#### UR

Lectura no confirmada

El nivel de aislamiento por omisión de la sentencia es el nivel de aislamiento del paquete en el que está enlazada la sentencia.

## Ejemplos

*Ejemplo 1:* Este ejemplo C pone el salario máximo de EMP en la variable del lenguaje principal MAXSALARY.

```
EXEC SQL SELECT MAX(SALARY)
 INTO :MAXSALARY
 FROM EMP;
```

*Ejemplo 2:* Este ejemplo C pone la fila para el empleado 528671, de EMP, en variables del lenguaje principal.

```
EXEC SQL SELECT * INTO :h1, :h2, :h3, :h4
 FROM EMP
 WHERE EMPNO = '528671';
```



## SET CONNECTION

La sentencia SET CONNECTION cambia el estado de una conexión de inactivo a actual, convirtiendo la ubicación especificada en el servidor actual. No está bajo control de transacción.

### Invocación

Aunque un recurso SQL interactivo pueda proporcionar una interfaz que dé la apariencia de ejecución interactiva, esta sentencia sólo puede incluirse dentro de un programa de aplicación. Se trata de una sentencia ejecutable que no puede prepararse de forma dinámica.

### Autorización

No se necesita ninguna.

### Sintaxis

```

▶▶ SET CONNECTION nombre-servidor
 └── variable-lengprinc ───▶

```

### Descripción

*nombre-servidor* o *variable-lengprinc*

Identifica el servidor de aplicaciones mediante el *nombre-servidor* especificado o una *variable-lengprinc* que contenga el *nombre-servidor*.

Si se especifica una *variable-lengprinc*, debe ser una variable de serie de caracteres con un atributo de longitud no superior a 8 y no debe contener una variable indicadora. El *nombre-servidor* contenido en la *variable-lengprinc* debe estar justificado por la izquierda y no estar delimitado por comillas.

Observe que el *nombre-servidor* es un seudónimo de base de datos que identifica al servidor de aplicaciones. Debe estar listado en el directorio local del petionario de aplicaciones.

El *nombre-servidor* o la *variable-lengprinc* debe identificar una conexión existente del proceso de aplicación. Si no identifican ninguna conexión existente, se genera un error (SQLSTATE 08003).

Si SET CONNECTION se aplica a la conexión actual, no se cambian los estados de ninguna de las conexiones del proceso de aplicación.

#### *Conexión satisfactoria*

Si la sentencia SET CONNECTION se ejecuta satisfactoriamente:

- No se realiza ninguna conexión. El registro especial CURRENT SERVER se actualiza con el *nombre-servidor* especificado.
- La conexión actual previa, si la hay, se coloca en el estado inactivo (suponiendo que se haya especificado un *nombre-servidor* distinto).

## SET CONNECTION

- El registro especial CURRENT SERVER y la SQLCA se actualizan de la misma manera que se documenta bajo los detalles de CONNECT Tipo 1 655.

### *Conexión no satisfactoria*

Si la sentencia SET CONNECTION falla:

- No importa cuál haya sido la razón de la anomalía, el estado de conexión del proceso de aplicación y los estados de sus conexiones permanecen invariables.
- Al igual que para un CONNECT de Tipo 1 no satisfactorio, el campo SQLERRP de la SQLCA se establece en el nombre del módulo que detectó el error.

### Notas

- La utilización de sentencias CONNECT de tipo 1 no excluye la utilización de SET CONNECTION, pero la sentencia fallará siempre (SQLSTATE 08003), a menos que la sentencia SET CONNECTION especifique la conexión actual, ya que no pueden existir conexiones inactivas.
- La opción de conexión SQLRULES(DB2) (consulte el apartado “Opciones que controlan la semántica de la unidad de trabajo distribuida” en la página 54) no excluye la utilización de SET CONNECTION, pero la sentencia no es necesaria porque se pueden utilizar en su lugar las sentencias CONNECT Tipo 2.
- Cuando se utiliza una conexión, se inactiva y después se restaura al estado actual en la misma unidad de trabajo, dicha conexión refleja su última utilización por el proceso de aplicación en relación al estado de bloques, cursores y sentencias preparadas.

### Ejemplos

Ejecute las sentencias de SQL de IBMSTHDB, ejecute las sentencias de SQL de IBMTOKDB y después ejecute más sentencias de SQL de IBMSTHDB.

```
EXEC SQL CONNECT TO IBMSTHDB;
/* Ejecuta las sentencias que hacen referencia a objetos de IBMSTHDB */
EXEC SQL CONNECT TO IBMTOKDB;
/* Ejecuta las sentencias que hacen referencia a objetos de IBMTOKDB */
EXEC SQL SET CONNECTION IBMSTHDB;
/* Ejecuta las sentencias que hacen referencia a objetos de IBMSTHDB */
```

Observe que la primera sentencia CONNECT crea la conexión IBMSTHDB, la segunda sentencia CONNECT lo coloca en el estado inactivo y la sentencia SET CONNECTION la devuelve al estado actual.

## SET CURRENT DEFAULT TRANSFORM GROUP

La sentencia SET CURRENT DEFAULT TRANSFORM GROUP cambia el valor del registro especial CURRENT DEFAULT TRANSFORM GROUP. La sentencia no está bajo el control de transacción.

### Invocación

Esta sentencia se puede intercalar en un programa de aplicación o se puede emitir utilizando sentencias de SQL dinámicas. Es una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

No se necesita ninguna autorización para ejecutar esta sentencia.

### Sintaxis

```

▶▶ SET CURRENT DEFAULT TRANSFORM GROUP = nombre-grupo

```

### Descripción

*nombre-grupo*

Especifica un nombre, formado por un solo elemento, que identifica un grupo de transformación definido para todos los tipos estructurados. Este nombre se puede utilizar en sentencias subsiguientes (o hasta que el valor del registro especial se vuelve a cambiar utilizando otra sentencia SET CURRENT DEFAULT TRANSFORM GROUP).

El nombre debe ser un identificador SQL, de hasta 18 caracteres de longitud (SQLSTATE 42815). Cuando se establece el registro especial, no se realiza ninguna validación de que el *nombre-grupo* esté definido para cualquier tipo estructurado. Sólo se comprueba la validez de la definición del grupo de transformación mencionado cuando se referencia explícitamente un tipo estructurado.

### Normas

- Si el valor especificado no se adapta a las reglas para un *nombre-grupo*, se emite un error (SQLSTATE 42815)
- Las funciones TO SQL y FROM SQL definidas en el grupo de transformación *nombre-grupo* se utilizan para intercambiar datos de tipo estructurado, definidos por el usuario, con un programa del lenguaje principal.

### Notas

- El valor inicial del registro especial CURRENT DEFAULT TRANSFORM GROUP es la serie de caracteres vacía.

## SET CURRENT DEFAULT TRANSFORM GROUP

- Vea “CURRENT DEFAULT TRANSFORM GROUP” en la página 134 para conocer más reglas referentes a la utilización del registro especial.

### Ejemplos

*Ejemplo 1:* Establecimiento del grupo de transformación por omisión en MYSTRUCT1. Este ejemplo utiliza las funciones TO SQL y FROM SQL definidas en el grupo de transformación MYSTRUCT1 para intercambiar variables de tipo estructurado, definidas por el usuario, con el programa actual del lenguaje principal.

```
SET CURRENT DEFAULT TRANSFORM GROUP = MYSTRUCT1
```

## SET CURRENT DEGREE

La sentencia SET CURRENT DEGREE asigna un valor al registro especial CURRENT DEGREE. La sentencia no está bajo el control de transacción.

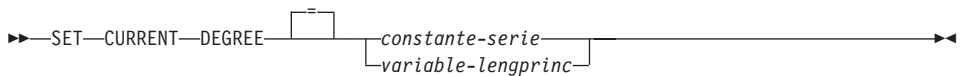
### Invocación

Esta sentencia se puede intercalar en un programa de aplicación o se puede emitir utilizando sentencias de SQL dinámicas. Es una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

No se necesita ninguna autorización para ejecutar esta sentencia.

### Sintaxis



### Descripción

El valor de CURRENT DEGREE se sustituye por el valor de la constante de serie o de la variable del lenguaje principal. El valor debe ser una serie de caracteres que no tenga una longitud superior a 5 bytes. El valor debe ser la representación de serie de caracteres de un entero entre 1 y 32.767 inclusive o 'ANY'.

Si el valor de CURRENT DEGREE representado como un entero es 1 cuando una sentencia de SQL se prepara dinámicamente, la ejecución de esta sentencia no utilizará el paralelismo intrapartición.

Si el valor de CURRENT DEGREE es un número cuando se prepara una sentencia de SQL dinámicamente, la ejecución de dicha sentencia puede implicar un paralelismo intrapartición con el grado especificado.

Si el valor de CURRENT DEGREE es 'ANY' cuando una sentencia de SQL se prepara dinámicamente, la ejecución de dicha sentencia puede implicar el paralelismo intrapartición que utiliza un grado determinado por el gestor de bases de datos.

#### *variable-lengprinc*

El tipo de datos de la *variable-lengprinc* debe ser CHAR o VARCHAR y su longitud no debe ser mayor que 5. Si el campo es más largo, se emite un error (SQLSTATE 42815). Si el valor real que se proporciona es mayor que el valor de sustitución especificado, la entrada se debe rellenar por la derecha con blancos. Los blancos iniciales no están permitidos (SQLSTATE 42815). Todos los valores de entrada se tratan como si no fuesen sensibles

## SET CURRENT DEGREE

a las mayúsculas y minúsculas. Si una *variable-length princ* tiene asociada una variable indicadora, el valor de dicha variable no debe indicar un valor nulo (SQLSTATE 42815).

*constante-serie*

La longitud de la *constante-serie* no debe exceder de 5.

### Notas

El grado de paralelismo intrapartición para las sentencias de SQL puede controlarse utilizando la opción DEGREE del mandato PREP o BIND. Consulte el manual *Consulta de mandatos* para obtener detalles sobre estos mandatos.

El grado de ejecución real del paralelismo intrapartición será inferior a:

- El parámetro de configuración del grado máximo de consulta (max\_querydegree)
- El grado de ejecución de la aplicación
- El grado de compilación de la sentencia de SQL

La configuración del gestor de bases de datos intraparelo debe estar activada para utilizar el paralelismo intrapartición. Si está desactivada, se pasará por alto el valor de este registro y la sentencia no utilizará el paralelismo intrapartición con el fin de optimización (SQLSTATE 01623).

Algunas sentencias de SQL no pueden utilizar el paralelismo intrapartición. Consulte el manual *Administration Guide* para una descripción del grado de paralelismo intrapartición y una lista de restricciones.

### Ejemplo

*Ejemplo 1:* La siguiente sentencia establece CURRENT DEGREE para que inhiba el paralelismo intrapartición.

```
SET CURRENT DEGREE = '1'
```

*Ejemplo 2:* La siguiente sentencia establece CURRENT DEGREE para que permita el paralelismo intrapartición.

```
SET CURRENT DEGREE = 'ANY'
```

## SET CURRENT EXPLAIN MODE

La sentencia SET CURRENT EXPLAIN MODE cambia el valor del registro especial CURRENT EXPLAIN MODE. No está bajo control de transacción.

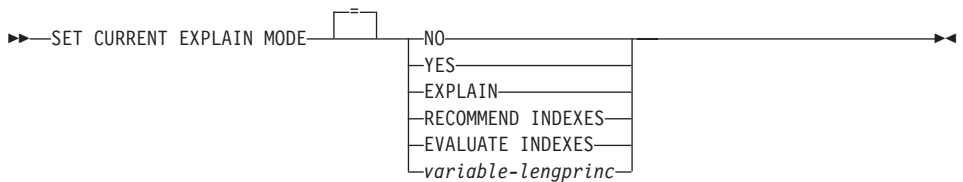
### Invocación

Esta sentencia puede incluirse en un programa de aplicación o emitirse de forma interactiva. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

No se necesita ninguna autorización especial para ejecutar esta sentencia.

### Sintaxis



### Descripción

#### NO

Inhabilita el recurso Explain. No se captura la información de Explain. NO es el valor inicial del registro especial.

#### YES

Habilita el recurso Explain y provoca que la información de Explain se inserte en las tablas de Explain para las sentencias de SQL dinámicas elegibles. Todas las sentencias de SQL dinámicas se compilan y ejecutan normalmente.

#### EXPLAIN

Habilita el recurso Explain y provoca que se capte la información de Explain para cualquier sentencia de SQL dinámica elegible que esté preparada. Sin embargo, no se ejecutan las sentencias dinámicas.

#### RECOMMEND INDEXES

Habilite el compilador SQL para los índices recomendados. Todas las peticiones que se ejecutan en esta modalidad Explain llenarán la tabla ADVISE\_INDEX con los índices recomendados. Así mismo, información de Explain será capturada en tablas de Explain para demostrar cómo se utilizan los índices recomendados, pero las sentencias no se compilan ni se ejecutan.

#### EVALUATE INDEXES

Habilita el compilador SQL para evaluar los índices. Los índices a evaluar se leen de la tabla ADVISE\_INDEX y deben marcarse con EVALUATE = Y.

## SET CURRENT EXPLAIN MODE

El optimizador genera índices virtuales basados en los valores de los catálogos. Todas las peticiones que se ejecutan en esta modalidad Explain serán compilados y optimizados utilizando estadísticas estimadas basadas en los índices virtuales. No se ejecutan las sentencias.

### *variable-lengprinc*

El tipo de datos de la *variable-lengprinc* debe ser CHAR o VARCHAR y su longitud no debe ser mayor que 254. Si se proporciona un campo más largo, se devolverá un error (SQLSTATE 42815). El valor especificado debe ser NO, YES, EXPLAIN, RECOMMEND INDEXES o EVALUATE INDEXES. Si el valor real que se proporciona es mayor que el valor de sustitución especificado, la entrada se debe rellenar por la derecha con blancos. Los blancos iniciales no están permitidos (SQLSTATE 42815). Todos los valores de entrada se tratan como si no fuesen sensibles a las mayúsculas y minúsculas. Si una *variable-lengprinc* tiene asociada una variable indicadora, el valor de dicha variable no debe indicar un valor nulo (SQLSTATE 42815).

## Notas

La información de Explain para las sentencias de SQL estáticas se puede capturar utilizando la opción EXPLAIN del mandato PREP o BIND. Si se especifica el valor ALL de la opción EXPLAIN y el valor de registro CURRENT EXPLAIN MODE es NO, se capturará la información de Explain para las sentencias de SQL dinámicas en tiempo de ejecución. Si el valor del registro CURRENT EXPLAIN MODE es distinto de NO, no se tiene en cuenta el valor de la opción de enlace EXPLAIN. Para obtener más información sobre la interacción entre la opción EXPLAIN y el registro especial CURRENT EXPLAIN MODE, consulte la Tabla 146 en la página 1496.

RECOMMEND INDEXES y EVALUATE INDEXES son modalidades especiales que sólo pueden establecerse con el mandato SET CURRENT EXPLAIN MODE. Estas modalidades no pueden establecerse utilizando las opciones PREP o BIND y no funcionan con el mandato SET CURRENT SNAPSHOT.

Si se activa el recurso Explain, el ID de autorización actual debe tener el privilegio INSERT para las tablas de Explain o se genera un error (SQLSTATE 42501).

Para obtener más información, consulte el manual *Administration Guide*.

## Ejemplo

*Ejemplo 1:* La sentencia siguiente establece el registro especial CURRENT EXPLAIN MODE, de modo que se capte información de Explain para cualquier sentencia de SQL dinámica elegible y la sentencia no se ejecutará.

```
SET CURRENT EXPLAIN MODE = EXPLAIN
```



## SET CURRENT EXPLAIN SNAPSHOT

La sentencia SET CURRENT EXPLAIN SNAPSHOT cambia el valor del registro especial CURRENT EXPLAIN SNAPSHOT. No está bajo control de transacción.

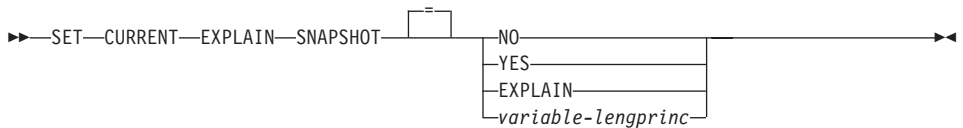
### Invocación

Esta sentencia puede incluirse en un programa de aplicación o emitirse de forma interactiva. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

No se necesita ninguna autorización para ejecutar esta sentencia.

### Sintaxis



### Descripción

#### NO

Inhabilita el servicio de instantánea de Explain. No se toma ninguna instantánea. NO es el valor inicial del registro especial.

#### YES

Habilita el recurso de instantánea de Explain, creando una instantánea de la representación interna de cada sentencia de SQL dinámica elegible. Esta información se inserta en la columna SNAPSHOT de la tabla EXPLAIN\_STATEMENT (consulte el “Apéndice K. Tablas de Explain y definiciones” en la página 1459).

El recurso EXPLAIN SNAPSHOT está pensado para utilizarlo con Visual Explain.

#### EXPLAIN

Habilita el recurso de instantánea de Explain, creando una instantánea de la representación interna de cada sentencia de SQL dinámica elegible que se prepara. Sin embargo, no se ejecutan las sentencias dinámicas.

#### *variable-lengprinc*

El tipo de datos de la *variable-lengprinc* debe ser CHAR o VARCHAR y la longitud de su contenido no debe ser mayor que 8. Si el campo es más largo, se emite un error (SQLSTATE 42815). El valor contenido en el registro debe ser NO, YES ni EXPLAIN. Si el valor real que se proporciona es mayor que el valor de sustitución especificado, la entrada se debe rellenar por la derecha con blancos. Los blancos iniciales no están

## SET CURRENT EXPLAIN SNAPSHOT

permitidos (SQLSTATE 42815). Todos los valores de entrada se tratan como si no fuesen sensibles a las mayúsculas y minúsculas. Si una *variable-lengprinc* tiene asociada una variable indicadora, el valor de dicha variable indicadora no debe indicar un valor nulo (SQLSTATE 42815).

### Notas

Las instantáneas de Explain para las sentencias de SQL estáticas se pueden capturar utilizando la opción EXPLSNAP del mandato PREP o BIND. Si se especifica el valor ALL de la opción EXPLSNAP y el valor del registro CURRENT EXPLAIN SNAPSHOT es NO, se capturarán las instantáneas de Explain para las sentencias de SQL dinámicas en tiempo de ejecución. Si el valor del registro CURRENT EXPLAIN SNAPSHOT es distinto de NO, se pasa por alto la opción EXPLSNAP. Para obtener más información sobre la interacción entre la opción EXPLSNAP y el registro especial CURRENT EXPLAIN SNAPSHOT, consulte la Tabla 147 en la página 1498.

Si se activa el recurso de instantánea de Explain, el ID de autorización actual debe tener el privilegio INSERT para las tablas de Explain o se genera un error (SQLSTATE 42501).

Para obtener más información, consulte el manual *Administration Guide*.

### Ejemplo

*Ejemplo 1:* La siguiente sentencia establece el registro especial CURRENT EXPLAIN SNAPSHOT de modo que se tomará una instantánea de Explain para cualquier sentencia de SQL dinámica elegible posterior y se ejecutará la sentencia.

```
SET CURRENT EXPLAIN SNAPSHOT = YES
```

*Ejemplo 2:* El ejemplo siguiente recupera el valor actual del registro especial CURRENT EXPLAIN SNAPSHOT en la variable del lenguaje principal llamada SNAP.

```
EXEC SQL VALUES (CURRENT EXPLAIN SNAPSHOT) INTO :SNAP;
```

## SET CURRENT PACKAGESET

La sentencia SET CURRENT PACKAGESET establece el nombre de esquema (identificador de colección) que se utilizará para seleccionar el paquete que se ha de utilizar para las sentencias de SQL posteriores. La sentencia no está bajo el control de transacción.

### Invocación

Esta sentencia sólo se puede incluir en un programa de aplicación. Se trata de una sentencia ejecutable que no puede prepararse de forma dinámica. Esta sentencia no está soportada en REXX.

### Autorización

No se necesita.

### Sintaxis

```

▶▶—SET—CURRENT PACKAGESET—=—constante-serie—
|—variable-lengprinc—▶▶

```

### Descripción

#### *constante-serie*

Es una constante de tipo serie con una longitud máxima de 30. Si es más larga que el valor máximo, se truncará durante la ejecución.

#### *variable-lengprinc*

Es una variable de tipo CHAR o VARCHAR con una longitud máxima de 30. No puede establecerse en nulo. Si tiene más del máximo, se truncará en tiempo de ejecución.

### Notas

- Esta sentencia permite que una aplicación especifique el nombre de esquema utilizado al seleccionar un paquete para una sentencia de SQL ejecutable. La sentencia se procesa en el cliente y no fluye al servidor de aplicaciones.
- Se puede utilizar la opción de enlace lógico COLLECTION para crear un paquete con un nombre de esquema especificado. Vea el manual *Consulta de mandatos* para conocer detalles.
- A diferencia de DB2 para MVS/ESA, la sentencia SET CURRENT PACKAGESET se implanta sin soporte para un registro especial llamado CURRENT PACKAGESET.

### Ejemplo

Suponga que el ID de usuario PRODUSA precompila la aplicación llamada TRYIT, estableciendo que 'PRODUSA' sea el nombre de esquema por omisión

## SET CURRENT PACKAGESET

del archivo de enlace lógico. Después, la aplicación se enlaza dos veces con diferentes opciones de enlace lógico. Se utilizan los siguientes mandatos del procesador de línea de mandatos:

```
DB2 CONNECT TO SAMPLE USER PRODUSA
DB2 BIND TRYIT.BND DATETIME USA
DB2 CONNECT TO SAMPLE USER PRODEUR
DB2 BIND TRYIT.BND DATETIME EUR COLLECTION 'PRODEUR'
```

Esto crea dos paquetes llamados TRYIT. El primer mandato de enlace lógico ha creado el paquete en el esquema llamado 'PRODUSA'. El segundo mandato de enlace lógico ha creado el paquete en el esquema llamado 'PRODEUR' basándose en la opción COLLECTION.

Suponga que la aplicación TRYIT contiene las sentencias siguientes:

```
EXEC SQL CONNECT TO SAMPLE;
.
.
EXEC SQL SELECT HIREDATE INTO :HD FROM EMPLOYEE WHERE EMPNO='000010'; 1
.
.
EXEC SQL SET CURRENT PACKAGESET 'PRODEUR'; 2
.
.
EXEC SQL SELECT HIREDATE INTO :HD FROM EMPLOYEE WHERE EMPNO='000010'; 3
```

- 1** Esta sentencia se ejecutará utilizando el paquete PRODUSA.TRYIT porque es el paquete por omisión para la aplicación. Por lo tanto, la fecha se devuelve en formato USA.
- 2** Esta sentencia establece el nombre de esquema en 'PRODEUR' para la selección del paquete.
- 3** Esta sentencia se ejecutará utilizando el paquete PRODEUR.TRYIT como resultado de la sentencia SET CURRENT PACKAGESET. Por lo tanto, la fecha se devuelve en formato EUR.

**SET CURRENT QUERY OPTIMIZATION**

La sentencia SET CURRENT QUERY OPTIMIZATION asigna un valor al registro especial CURRENT QUERY OPTIMIZATION. El valor especifica la clase actual de técnicas de optimización habilitadas al preparar las sentencias de SQL dinámicas. No está bajo control de transacción.

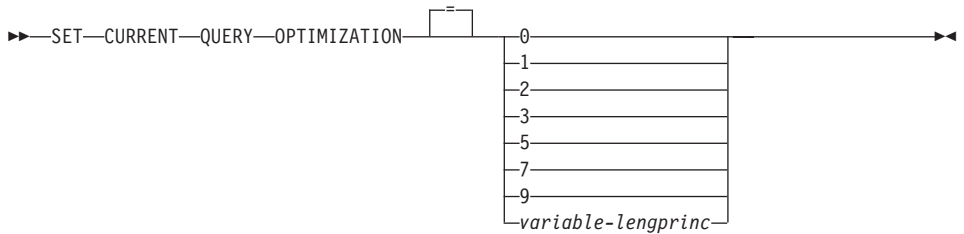
**Invocación**

Esta sentencia se puede intercalar en un programa de aplicación o se puede emitir utilizando sentencias de SQL dinámicas. Es una sentencia ejecutable que puede prepararse de forma dinámica.

**Autorización**

No se necesita ninguna autorización para ejecutar esta sentencia.

**Sintaxis**



**Descripción**

*clase-optimización*

La *clase-optimización* se puede especificar como una constante de enteros o como el nombre de una variable del lenguaje principal que contendrá el valor adecuado en tiempo de ejecución. A continuación encontrará una visión general de las clases (para obtener detalles consulte el manual *Administration Guide* ).

- 0 Especifica que se efectúa una optimización mínima para generar un plan de acceso. Esta clase es la más adecuada para el acceso SQL dinámico simple para tablas bien indexadas.
- 1 Especifica que se efectúa una optimización más o menos comparable a DB2 Versión 1 para generar un plan de acceso.
- 2 Especifica un nivel de optimización superior al de DB2 Versión 1, pero con un coste de optimización significativamente menor que los niveles 3 y superiores, especialmente para consultas muy complejas.

## SET CURRENT QUERY OPTIMIZATION

- 3 Especifica que se efectúa una optimización moderada para generar un plan de acceso.
- 5 Especifica que se efectúa una optimización significativa para generar un plan de acceso. Para consultas SQL dinámicas complejas, se utilizan las reglas heurísticas para limitar el tiempo empleado en seleccionar un plan de acceso. Cuando sea posible, las consultas utilizarán tablas de resumen en lugar de las tablas base principales.
- 7 Especifica que se efectúa una optimización significativa para generar un plan de acceso. Similar al 5 pero sin las reglas heurísticas.
- 9 Especifica que se efectúa una optimización máxima para generar un plan de acceso. Puede expandir mucho el número de planes de acceso posibles que se evalúan. Esta clase debe utilizarse para determinar si se puede generar un plan de acceso mejor para las consultas muy complejas y de muy larga ejecución que utilizan tablas grandes. Las medidas de explicación y de rendimiento se pueden utilizar para verificar que se haya generado el plan mejor.

### *variable-lengprinc*

El tipo de datos es INTEGER. El valor debe estar en el rango de 0 a 9 (SQLSTATE 42815) pero debe ser 0, 1, 2,3, 5, 7 ó 9 (SQLSTATE 01608). Si una *variable-lengprinc* tiene asociada una variable indicadora, el valor de dicha variable indicadora no debe indicar un valor nulo (SQLSTATE 42815).

## Notas

- Cuando el registro CURRENT QUERY OPTIMIZATION se establece en un valor en particular, se habilita un conjunto de reglas para volver a escribir la consulta y ciertas variables de optimización toman valores determinados. Esta clase de técnicas de optimización se utiliza después durante la preparación de sentencias de SQL.
- En general, el cambio de la clase de optimización afecta al tiempo de ejecución de la aplicación, al tiempo de compilación y a los recursos necesarios. La mayoría de sentencias se optimizarán de manera adecuada utilizando la clase de optimización de consulta por omisión. Las clases de optimización de consulta inferiores, especialmente las clases 1 y 2, pueden ser adecuadas para las sentencias de SQL dinámicas para las cuales los recursos que consume una operación PREPARE dinámica son una parte significativa de los necesarios para ejecutar la consulta. Las clases de optimización superiores sólo deben elegirse después de tener en cuenta los recursos adicionales que pueden consumirse y verificar que se haya

generado un plan de acceso mejor. Para obtener detalles adicionales sobre el funcionamiento asociado con cada clase de optimización de consulta, vea el manual *Administration Guide*.

- Las clases de optimización de consulta deben estar en el rango de 0 a 9. Las clases fuera de este rango devolverán un error (SQLSTATE 42815). Las clases no soportadas dentro de este rango devolverán un aviso (SQLSTATE 01608) y se sustituirán por la siguiente clase de optimización de consulta inferior. Por ejemplo, una clase 6 de optimización de consulta se sustituirá por 5.
- Las sentencias preparadas dinámicamente utilizan la clase de optimización que se ha establecido por la sentencia SET CURRENT QUERY OPTIMIZATION más reciente que se ha ejecutado. En las clases en que no se ha ejecutado todavía una sentencia SET CURRENT QUERY OPTIMIZATION, la clase de optimización de consulta se determina por el valor del parámetro de configuración de la base de datos, `dft_queryopt`.
- Las sentencias enlazadas estáticamente no utilizan el registro especial CURRENT QUERY OPTIMIZATION; por lo tanto, esta sentencia no tiene ningún efecto sobre ellas. La opción QUERYOPT se utiliza durante el preproceso o enlace lógico para especificar la clase de optimización deseada para las sentencias enlazadas estáticamente. Si no se especifica QUERYOPT, se utiliza el valor por omisión especificado por el parámetro de configuración de la base de datos, `dft_queryopt`. Consulte el mandato BIND en el manual *Consulta de mandatos* para obtener detalles.
- No se retrotrae el resultado de la ejecución de la sentencia SET CURRENT QUERY OPTIMIZATION si se retrotrae la unidad de trabajo en la que se ejecuta.

### Ejemplos

*Ejemplo 1:* Este ejemplo muestra cómo se puede seleccionar el grado de optimización más alto.

```
SET CURRENT QUERY OPTIMIZATION 9
```

*Ejemplo 2:* El ejemplo siguiente muestra cómo se puede utilizar el registro especial CURRENT QUERY OPTIMIZATION en una consulta.

Utilizando la vista de catálogo SYSCAT.PACKAGES, encuentre todos los planes que se han enlazado con el mismo valor que el valor actual del registro especial CURRENT QUERY OPTIMIZATION.

```
EXEC SQL DECLARE C1 CURSOR FOR
SELECT PKGNAME, PKGSHEMA FROM SYSCAT.PACKAGES
WHERE QUERYOPT = CURRENT QUERY OPTIMIZATION
```

## SET CURRENT REFRESH AGE

---

### SET CURRENT REFRESH AGE

La sentencia SET CURRENT REFRESH AGE cambia el valor del registro especial CURRENT REFRESH AGE. No está bajo control de transacción.

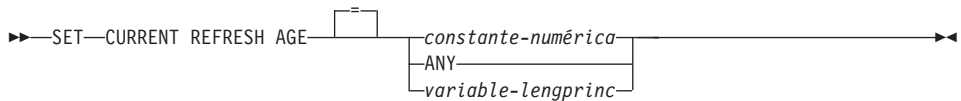
#### Invocación

Esta sentencia puede incluirse en un programa de aplicación o emitirse de forma interactiva. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

#### Autorización

No se necesita ninguna autorización para ejecutar esta sentencia.

#### Sintaxis



#### Descripción

##### *constante-numérica*

Un valor DECIMAL(20,6) que representa una duración de indicación de la hora. El valor debe ser 0 o 99 999 999 999 999 (la parte de microsegundos del valor se ignora y, por lo tanto, puede ser cualquier valor).

**0**

Indica que sólo pueden utilizarse las tablas de resumen definidas con REFRESH IMMEDIATE para optimizar el proceso de una consulta.

**999999999999999**

Indica que puede utilizarse cualquier tabla de resumen definida con REFRESH DEFERRED o REFRESH IMMEDIATE para optimizar el proceso de una consulta. Este valor representa 9 999 años, 99 meses, 99 días, 99 horas, 99 minutos y 99 segundos.

##### **ANY**

Es una abreviación de 999999999999999.

##### *variable-lengprinc*

Una variable de tipo DECIMAL(20,6) u otro tipo asignable a DECIMAL(20,6). No puede establecerse en nulo. Si una *variable-lengprinc* tiene asociada una variable indicadora, el valor de dicha variable indicadora no debe indicar un valor nulo (SQLSTATE 42815). El valor de la *variable-lengprinc* debe ser 0 ó 99 999 999 999 999.000000.



## Notas

- El valor inicial del registro especial CURRENT REFRESH AGE es cero.
- Debe tener cuidado cuando establezca el registro especial CURRENT REFRESH AGE en un valor que no sea cero. Si se permite una tabla de resumen que puede que no represente los valores de la tabla base principal que se utilizará para optimizar el proceso de la consulta, es posible que el resultado de la consulta NO represente con exactitud los datos de la tabla principal. Esto puede ser razonable cuando sepa que los datos principales no han cambiado o esté dispuesto a aceptar el grado de error del resultado basándose en el conocimiento de los datos.
- El valor CURRENT REFRESH AGE de 99 999 999 999 999 no puede utilizarse en operaciones aritméticas de indicación de la hora porque el resultado estaría fuera del rango válido de fechas (SQLSTATE 22008).

## Ejemplos

*Ejemplo 1:* La sentencia siguiente establece el registro especial CURRENT REFRESH AGE.

```
SET CURRENT REFRESH AGE ANY
```

*Ejemplo 2:*

El ejemplo siguiente recupera el valor actual del registro especial CURRENT REFRESH AGE en la variable del lenguaje principal denominada CURMAXAGE.

```
EXEC SQL VALUES (CURRENT REFRESH AGE) INTO :CURMAXAGE;
```

El valor sería 99999999999999,000000, establecido en el ejemplo anterior.

## SET ENCRYPTION PASSWORD

---

### SET ENCRYPTION PASSWORD

La sentencia SET ENCRYPTION PASSWORD establece la contraseña que utilizarán las funciones ENCRYPT, DECRYPT\_BIN y DECRYPT\_CHAR. La contraseña no está vinculada a la autenticación de DB2 y se utiliza solamente para el cifrado de datos.

La sentencia no está bajo el control de la transacción.

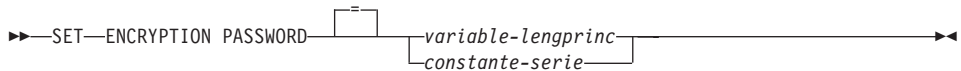
#### Invocación

La sentencia puede incluirse en un programa de aplicación o emitirse interactivamente. Se trata de una sentencia ejecutable que puede prepararse dinámicamente.

#### Autorización

No se necesita ninguna autorización para ejecutar esta sentencia.

#### Sintaxis



#### Descripción

Las funciones incorporadas ENCRYPT, DECRYPT\_BIN y DECRYPT\_CHAR pueden utilizar ENCRYPTION PASSWORD para el cifrado basado en contraseña. La longitud debe ser de 6 a 127 bytes. Todos los caracteres deben especificarse en las mayúsculas y minúsculas exactas que se desean porque no se realiza ninguna conversión automática a caracteres en mayúsculas.

##### *variable-lengprinc*

Una variable de tipo CHAR o VARCHAR. La longitud de *variable-lengprinc* debe tener una longitud de 6 a 127 bytes (SQLSTATE 428FC). No se puede establecer en nulo. Todos los caracteres se especifican en las mayúsculas y minúsculas que se desean porque no se realiza ninguna conversión a caracteres en mayúsculas.

##### *constante-serie*

Constante de serie de caracteres. La longitud debe ser de 6 a 127 bytes (SQLSTATE 428FC).

#### Notas

- El valor inicial de ENCRYPTION PASSWORD es la serie vacía (' ').
- La *variable-lengprinc* o *constante-serie* se transmite al servidor de bases de datos utilizando los mecanismos normales de DB2.

## SET ENCRYPTION PASSWORD

- Consulte el apartado “ENCRYPT” en la página 331 y el apartado “DECRYPT\_BIN y DECRYPT\_CHAR” en la página 314 si desea información adicional sobre la utilización de esta sentencia.

### Ejemplos

*Ejemplo 1:* La sentencia siguiente establece la contraseña de cifrado (ENCRYPTION PASSWORD).

```
SET ENCRYPTION PASSWORD = 'Gre89Ea'
```

## SET EVENT MONITOR STATE

---

### SET EVENT MONITOR STATE

La sentencia SET EVENT MONITOR STATE activa o desactiva el supervisor de sucesos. El estado actual de un supervisor de sucesos (activo o inactivo) se determina utilizando la función incorporada EVENT\_MON\_STATE. La sentencia SET EVENT MONITOR STATE no está bajo el control de transacción.

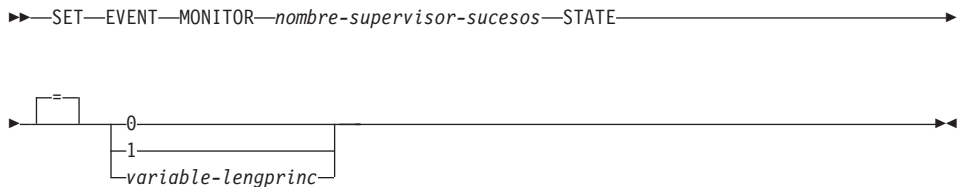
#### Invocación

Esta sentencia se puede intercalar en un programa de aplicación o se puede emitir utilizando sentencias de SQL dinámicas. Es una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

#### Autorización

El ID de autorización de la sentencia debe tener la autorización SYSADM o DBADM (SQLSTATE 42815).

#### Sintaxis



#### Descripción

##### *nombre-supervisor-sucesos*

Identifica el supervisor de sucesos que se ha de activar o desactivar. El nombre debe identificar un supervisor de sucesos que exista en el catálogo (SQLSTATE 42704).

##### *estado-nuevo*

El *estado-nuevo* puede especificarse como una constante de enteros o como el nombre de una variable del lenguaje principal que contendrá el valor adecuado en tiempo de ejecución. Se puede especificar lo siguiente:

- 0 Indica que el supervisor de sucesos especificado debe desactivarse.
- 1 Indica que el supervisor de sucesos especificado debe activarse. El supervisor de sucesos no debe estar activo todavía; de lo contrario se emite un aviso (SQLSTATE 01598).

*variable-lengprinc*

El tipo de datos es INTEGER. El valor especificado debe ser 0 ó 1 (SQLSTATE 42815). Si una *variable-lengprinc* tiene asociada una variable indicadora, el valor de dicha variable indicadora no debe indicar un valor nulo (SQLSTATE 42815).

**Normas**

- Aunque puede definirse un número no limitado de supervisores de sucesos, existe un límite de 32 supervisores de sucesos que pueden estar activos simultáneamente (SQLSTATE 54030).
- Para activar un supervisor de sucesos, la transacción en la que se ha creado el supervisor de sucesos debe haberse confirmado (SQLSTATE 55033). Esta regla impide (en una unidad de trabajo) la creación de un supervisor de sucesos, la activación del supervisor y después la retrotracción de la transacción.
- Si el número o tamaño de los archivos del supervisor de sucesos excede de los valores especificados para MAXFILES o MAXFILESIZE en la sentencia CREATE EVENT MONITOR, se genera un error (SQLSTATE 54031).
- Si la vía de acceso de destino del supervisor de sucesos (que se ha especificado en la sentencia CREATE EVENT MONITOR) ya se utiliza en otro supervisor de sucesos, se genera un error (SQLSTATE 51026).

**Notas**

- La activación de un supervisor de sucesos realiza una restauración de cualquier contador asociado al mismo.

**Ejemplo**

El ejemplo siguiente activa un supervisor de sucesos llamado SMITHPAY.

```
SET EVENT MONITOR SMITHPAY STATE = 1
```

---

### SET INTEGRITY

La sentencia SET INTEGRITY<sup>106</sup> se utiliza para uno de los fines siguientes:

- Desactivar la comprobación de la integridad para una o más tablas. Esto incluye la restricción de comprobación y el control de restricción de referencia, el control de la integridad de los enlaces de datos y la generación de valores para columnas generadas. Si la tabla es una tabla de resumen con REFRESH IMMEDIATE, se desactiva la renovación inmediata de los datos. Observe que esto coloca la tabla o tablas en un *estado de comprobación pendiente* en el que sólo se permite un acceso limitado por un conjunto restringido de sentencias y mandatos. Se continúan comprobando las restricciones de unicidad y de clave primaria.
- Volver a activar la comprobación de integridad en una o varias tablas y llevar a cabo toda la comprobación diferida. Si la tabla es una tabla de resumen, los datos se renuevan de la manera necesaria y, cuando está definida con el atributo REFRESH IMMEDIATE, se activa la renovación inmediata de los datos.
- Activar la comprobación de integridad para una o más tablas sin realizar primero ninguna comprobación de integridad diferida. Si la tabla es una tabla de resumen definida con el atributo REFRESH IMMEDIATE, se activa la renovación inmediata de los datos.
- Colocar la tabla en estado de comprobación pendiente si la tabla ya se encuentra en estado pendiente de reconciliación de DataLink (DRP) o en estado de reconciliación de DataLink no posible (DRNP). Si una tabla no se encuentra en ninguno de esos dos estados, establezca la tabla incondicionalmente en el estado DRP y el estado de comprobación pendiente.

Cuando la sentencia se utiliza para comprobar la integridad para una tabla después de que se haya cargado, el sistema procesará por omisión y de manera incremental la tabla mediante la comprobación sólo de la parte añadida para violaciones de restricción. Sin embargo, existen algunas situaciones en las que el sistema determina que es necesario el proceso completo (para ello comprueba si en la tabla hay violaciones de restricción) a fin de asegurar la integridad de los datos. Existe también una situación en la que el usuario necesita pedir de manera explícita el proceso incremental mediante la especificación de la opción INCREMENTAL. Vea “Notas” en la página 1174 para obtener detalles.

La sentencia SET INTEGRITY está bajo control de transacciones.

---

106. La sentencia SET INTEGRITY, en lugar de la sentencia SET CONSTRAINTS, es el método preferido para trabajar con la comprobación de la integridad en DB2.

## Invocación

Esta sentencia se puede intercalar en un programa de aplicación o se puede emitir utilizando sentencias de SQL dinámicas. Es una sentencia ejecutable que puede prepararse de forma dinámica. Sin embargo, si es aplicable la opción de enlace DYNAMICRULES BIND, la sentencia no se puede preparar de forma dinámica (SQLSTATE 42509).

## Autorización

Los privilegios necesarios para ejecutar SET INTEGRITY dependen de la utilización de la sentencia, tal como se resalta abajo:

1. Desactivar la comprobación de la integridad.

El ID de autorización de la sentencia debe tener como mínimo uno de los privilegios siguientes:

- Privilegio CONTROL para las tablas y todos sus objetos dependientes y descendientes en las restricciones de integridad de referencia.
- Autorización SYSADM o DBADM
- Autorización LOAD

2. Activar la comprobación de la integridad y lleve a cabo la comprobación.

El ID de autorización de la sentencia debe tener como mínimo uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Privilegio CONTROL para las tablas que se están comprobando y, si se anotan excepciones en una o más tablas, privilegio INSERT para las tablas de excepciones.
- Autorización LOAD y, si se anotan excepciones en una o más tablas:
  - Privilegio SELECT y DELETE para cada tabla que se comprueba; y
  - Privilegio INSERT para las tablas de excepciones.

3. Activar las restricciones de integridad sin antes llevar a cabo la comprobación.

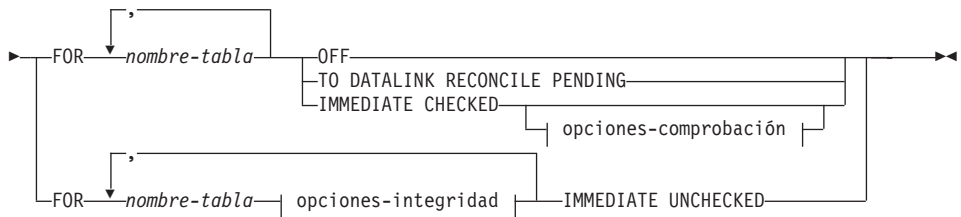
El ID de autorización de la sentencia debe tener como mínimo uno de los siguientes:

- Autorización SYSADM o DBADM
- Privilegio CONTROL para las tablas que se están comprobando
- Autorización LOAD

## Sintaxis

►►—SET—INTEGRITY—<sup>(1)</sup>—————►

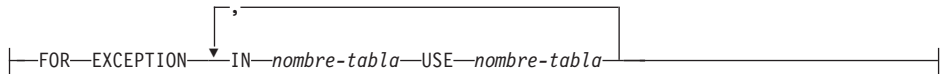
## SET INTEGRITY



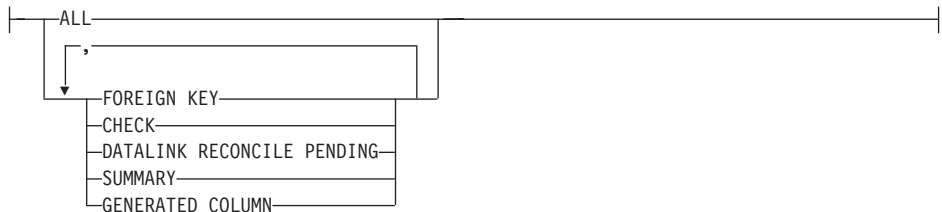
### opciones-comprobación:



### cláusula-excepción:



### opciones-integridad:



### Notas:

- 1 La palabra clave CONSTRAINTS continuará siendo soportada para la compatibilidad con versiones anteriores.

## Descripción

### *nombre-tabla*

Identifica una tabla para el proceso de la integridad. Debe ser una tabla descrita en el catálogo y no debe ser una vista, una tabla del catálogo ni una tabla con tipo.

### OFF

Especifica que las tablas deben tener desactivadas sus restricciones de clave foránea, restricciones de comprobación y la generación de columnas y, por lo tanto, deben colocarse en el estado de comprobación pendiente.



Si es una tabla de resumen, se desactiva la renovación inmediata (si es aplicable) y la tabla de resumen se coloca en estado de comprobación pendiente.

Observe que es posible que una tabla pueda estar ya en estado de comprobación pendiente con sólo un tipo de comprobación de integridad desactivada; en dicha situación, el otro tipo de comprobación de integridad también se desactivará.

Si una tabla de la lista es una tabla padre, el estado de comprobación pendiente para las restricciones de clave foránea se extiende a todas las tablas dependientes y descendientes.

Si una tabla de la lista es la tabla principal de una tabla de resumen, el estado de comprobación pendiente se extiende a esta tabla de resumen.

Sólo está permitida una actividad limitada en la tabla que está en el estado de comprobación pendiente. El apartado "Notas" en la página 1174 lista las restricciones.

#### **TO DATALINK RECONCILE PENDING**

Especifica que las tablas van a tener desactivada la comprobación de las restricciones de integridad de DATALINK y las tablas se van a colocar en el estado de comprobación pendiente. Si la tabla ya se encuentra en el estado de reconciliación de DataLink no posible (DRNP), permanecerá en este estado y en el de pendiente de comprobación. De lo contrario, la tabla se establecerá en un estado pendiente de reconciliación de DataLink (DRP).

La tabla dependiente y la descendiente no quedan afectadas cuando se especifica esta opción.

#### **IMMEDIATE CHECKED**

Especifica que la tabla ha de tener activadas sus restricciones y que se ha de llevar a cabo la comprobación de integridad que se ha diferido. Esto se efectúa de acuerdo con la información establecida en las columnas STATUS y CONST\_CHECKED del catálogo SYSCAT.TABLES. Es decir:

- El valor de STATUS debe ser C (la tabla está en estado de comprobación pendiente) o se devuelve un error (SQLSTATE 51027).
- El valor de CONST\_CHECKED indica qué opciones de integridad se han de comprobar.

Si es una tabla de resumen, los datos se comprueban con la consulta y se renuevan de la manera necesaria.

Los valores DATALINK no se comprueban aunque la tabla esté en estado DRP o DRNP. Debe utilizarse la API o el mandato RECONCILE para llevar a cabo la reconciliación de valores DATALINK. La tabla dejará de encontrarse en estado de comprobación pendiente, pero continuará

## SET INTEGRITY

teniendo establecido el distintivo DRP o DRNP. Esto hace que la tabla sea utilizable mientras la reconciliación de valores DATALINK se puede diferir a otro momento.

### *opciones-comprobación*

#### **FORCE GENERATED**

Si la tabla contiene columnas generadas, los valores se calculan de acuerdo con la expresión y se guardan en la columna. Si no se especifica esta cláusula, los valores actuales se comparan con el valor calculado de la expresión como si existiera una restricción de comprobación de igualdad.

#### **INCREMENTAL**

Especifica la aplicación de las comprobaciones de integridad diferida en la parte añadida (si hay alguna) de la tabla. Si no se puede llevar a cabo esta petición (por ejemplo, el sistema detecta que toda la tabla necesita ser comprobada para la integridad de los datos), se devolverá un error (SQLSTATE 55019). Si no se especifica el atributo, el sistema determinará si es posible llevar a cabo un proceso incremental; si no es posible, toda la tabla será comprobada. Consulte las Notas para conocer las situaciones en las que el sistema elige el proceso completo (comprobación de toda la tabla para asegurar su integridad) en lugar del proceso parcial. Consulte también en las Notas los casos en que es necesaria la opción INCREMENTAL y los casos en los que no se puede especificar.

Si la tabla no está en el estado de comprobación pendiente, se devolverá un error (SQLSTATE 55019).

### *cláusula-excepción*

#### **FOR EXCEPTION**

Indica que cualquier fila que viole una restricción de clave foránea o una restricción de comprobación se copiará en una tabla de excepciones y se suprimirá de la tabla original. Consulte el “Apéndice N. Tablas de excepciones” en la página 1505 para obtener más información sobre estas tablas definidas por el usuario. Incluso si se detectan errores, se vuelven a activar las restricciones de nuevo y la tabla sale del estado de comprobación pendiente. Se emite un aviso (SQLSTATE 01603) para indicar que se ha movido una o varias filas a las tablas de excepciones.

Si no se especifica la cláusula FOR EXCEPTION y se viola alguna restricción, entonces sólo se devuelve al usuario la primera violación (SQLSTATE 23514). En el caso de una violación en cualquier tabla, todas las tablas se dejan en el estado de comprobación pendiente, tal como estaban antes de la ejecución

de la sentencia. Esta cláusula no puede especificarse si *nombre-tabla* es una tabla de resumen (SQLSTATE 42997).

**IN** *nombre-tabla*

Especifica la tabla de la que se han de copiar las filas que violan restricciones. Debe haber una tabla de excepciones especificada para cada tabla que se comprueba.

**USE** *nombre-tabla*

Especifica la tabla de excepciones en la que se han de copiar las filas erróneas.

*opciones-integridad*

Se utiliza para definir las opciones de integridad que se establecen en IMMEDIATE UNCHECKED.

**ALL**

Indica que se van a activar todas las restricciones de integridad.

**FOREIGN KEY**

Indica que se han de activar las restricciones de clave foránea.

**CHECK**

Indica que se han de activar las restricciones de comprobación.

**DATALINK RECONCILE PENDING**

Indica que se van a activar las restricciones de integridad de DATALINK.

**SUMMARY**

Indica que se debe activar la renovación inmediata para una tabla de resumen con el atributo REFRESH IMMEDIATE.

**GENERATED COLUMN**

Indica que se deben activar columnas generadas.

**IMMEDIATE UNCHECKED**

Especifica uno de los siguientes:

- La tabla ha de tener activadas sus comprobaciones de integridad (y, por lo tanto, han de salir del estado de comprobación pendiente) sin comprobarse las violaciones de integridad o bien la tabla de resumen ha de tener activada la renovación inmediata y ha de salir del estado de comprobación pendiente.

Esto se indica para una tabla determinada especificando ALL o especificando CHECK cuando sólo se desactiven las restricciones de comprobación para la tabla, o especificando FOREIGN KEY cuando sólo se desactiven las restricciones de clave foránea, o especificando DATALINK RECONCILE PENDING cuando sólo se desactiven las restricciones de integridad de DATALINK para la tabla, o bien

## SET INTEGRITY

especificando SUMMARY cuando sólo se desactive la comprobación de consulta de tabla de resumen para esa tabla.

- La tabla debe tener activado un tipo de comprobación de la integridad, pero se debe dejar en el estado de comprobación pendiente.  
Esto se indica para una tabla determinada especificando solamente CHECK, FOREIGN KEY, SUMMARY, GENERATED COLUMN o DATALINK RECONCILE PENDING cuando se desactiva cualquiera de estos tipos de restricciones para esa tabla.

El cambio de estado no se amplía a las tablas que no se incluyan explícitamente en la lista.

Si la tabla padre de otra dependiente está en el estado de comprobación pendiente, no se pueden marcar las restricciones de clave foránea de una tabla dependiente para que eviten la comprobación (la comprobación de las restricciones de comprobación puede evitarse).

Deben tenerse en cuenta las implicaciones con respecto a la integridad de los datos antes de utilizar esta opción. Vea “Notas”.

### Notas

- Efectos en las tablas en el estado de comprobación pendiente:
  - La utilización de SELECT, INSERT, UPDATE o DELETE se inhabilita en una tabla que esté:
    - en el estado de comprobación pendiente
    - o necesite acceder a otra tabla que esté en el estado de comprobación pendiente.

Por ejemplo, no está permitida la operación DELETE de una fila de una tabla padre en cascada para una tabla dependiente que está en el estado de comprobación pendiente.

- Normalmente las nuevas restricciones añadidas a una tabla se imponen inmediatamente. Sin embargo, si la tabla está en estado de comprobación pendiente, la comprobación de cualquier nueva restricción se difiere hasta que la tabla sale del estado de comprobación pendiente.
- La sentencia CREATE INDEX no puede hacer referencia a ninguna tabla que esté en estado de comprobación pendiente. De manera similar, ALTER TABLE para añadir una clave primaria o una restricción de unicidad no puede hacer referencia a ninguna tabla que esté en estado de comprobación pendiente.
- Los programas de utilidad EXPORT, IMPORT, REORG y REORGCHK no tienen permitido funcionar en una tabla en el estado de comprobación

pendiente. Observe que el programa de utilidad IMPORT difiere del programa de utilidad LOAD en que siempre comprueba las restricciones inmediatamente.

- Los programas de utilidad LOAD, BACKUP, RESTORE, ROLLFORWARD, UPDATE STATISTICS, RUNSTATS, LIST HISTORY y ROLLFORWARD están permitidos en una tabla en el estado de comprobación pendiente.
- Las sentencias ALTER TABLE, COMMENT ON, DROP TABLE, CREATE ALIAS, CREATE TRIGGER, CREATE VIEW, GRANT, REVOKE y SET INTEGRITY pueden hacer referencia a una tabla que esté en estado de comprobación pendiente.
- Los paquetes, vistas y cualquier otro objeto que dependa de una tabla que está en estado de comprobación pendiente devolverá un error cuando se acceda la tabla en tiempo de ejecución.

La eliminación de las filas con las violaciones mediante la sentencia SET INTEGRITY no es un suceso de supresión. Por lo tanto, una sentencia SET INTEGRITY nunca activa los desencadenantes. Similarmente, la actualización de columnas mediante la opción FORCE GENERATED no produce la activación de desencadenantes.

- Debido a que el proceso incremental es la conducta por omisión, la opción INCREMENTAL no es necesaria en la mayoría de los casos. Sin embargo, es necesaria en dos casos:
  - Para forzar el proceso incremental en una tabla que había salido del estado de comprobación pendiente con anterioridad mediante la opción IMMEDIATE UNCHECKED. Por omisión, el sistema elige el proceso total para verificar la integridad de TODOS los datos. Esta conducta por omisión puede alterarse temporalmente especificando la opción INCREMENTAL para comprobar únicamente la nueva parte añadida. (Consulte el apartado "Aviso acerca de la utilización de la cláusula IMMEDIATE UNCHECKED" para obtener más detalles.)
  - Para asegurar que las comprobaciones de integridad son realmente procesadas de forma incremental. Al especificar la opción INCREMENTAL, el sistema devuelve un error (SQLSTATE 55019) cuando el sistema detecta que es necesario el proceso total para asegurar la integridad de los datos.
- Aviso acerca de la utilización de la cláusula IMMEDIATE UNCHECKED:
  - Esta cláusula está pensada para que la utilicen programas de utilidad y no es aconsejable que la utilicen los programas de aplicación. El hecho que se haya activado la comprobación de integridad sin realizar una comprobación diferida se registrará en el catálogo (el valor de la columna CONST\_CHECKED de la vista SYSCAT.TABLES se establecerá

## SET INTEGRITY

en 'U'). Esto indica que el usuario ha asumido la tarea de asegurar la integridad de los datos con respecto a las restricciones específicas. Este valor permanece en vigor hasta que:

- La tabla se vuelve a poner en estado de comprobación pendiente (haciendo referencia a la tabla en una sentencia SET INTEGRITY con la cláusula OFF), en cuyo momento los valores 'U' de la columna CONST\_CHECKED pasarán a ser valores 'W', lo que indica que el usuario había asumido anteriormente la tarea de asegurar la integridad de los datos y el sistema necesita verificar los datos.
- Se eliminan todas las restricciones no comprobadas para la tabla.
- Se emite una sentencia REFRESH TABLE para una tabla de resumen.

El estado 'W' se diferencia del estado 'N' en que registra el hecho de que la integridad ha sido comprobada por el usuario, pero no todavía por el sistema, y si se le da la opción, el sistema vuelve a comprobar la integridad de toda la tabla y luego establece el estado en 'Y'. Si no se proporciona ninguna una opción (por ejemplo, cuando se especifica IMMEDIATE UNCHECKED o INCREMENTAL), se vuelve a cambiar al estado 'U' para registrar el hecho de que algunos datos están todavía sin verificar por el sistema. En el último caso (INCREMENTAL), se devuelve un aviso (SQLSTATE 01636).

- Después de añadir datos utilizado Cargar insertar, la sentencia SET INTEGRITY ... IMMEDIATE CHECKED comprueba la tabla para restricciones de violación y entonces saca la tabla del estado de comprobación pendiente. El sistema determina si es posible procesar la tabla de modo incremental. Si es posible, sólo se comprueba la parte añadida para las violaciones de integridad. Si no es posible, el sistema comprobará toda la tabla para las violaciones de integridad (vea a continuación las situaciones en las que el sistema elige el proceso total).
- Estas son las situaciones en las que el sistema comprueba la integridad de toda la tabla si el usuario no especifica la opción INCREMENTAL en la sentencia SET INTEGRITY para la tabla T definida con IMMEDIATE CHECKED:
  1. cuando la tabla T tiene un o más valores 'W' en su columna CONST\_CHECKED en el catálogo SYSCAT.TABLES.
- Estas son las situaciones en las que el sistema debe comprobar la integridad de toda la tabla (no se puede especificar la opción INCREMENTAL) para la sentencia SET INTEGRITY de la tabla T definida con IMMEDIATE CHECKED:
  1. cuando se han añadido nuevas restricciones a T o a cualquiera de sus tablas padre que están en estado de comprobación pendiente
  2. cuando ha tenido lugar una carga con sustitución en T, o se ha activado la opción NOT LOGGED INITIALLY WITH EMPTY TABLE después de la última comprobación de integridad en T

3. (efecto de propagación en cascada del proceso total) cuando ha tenido lugar una carga con sustitución o una comprobación de integridad no incremental para cualquier tabla padre de T
  4. Si la tabla estaba en estado de comprobación pendiente antes de la migración, es necesario el proceso total la primera vez que se comprueba la integridad de la tabla después de la migración
  5. si ha tenido lugar una recuperación de datos para el espacio de tablas donde reside la tabla o su tabla padre.
- Una tabla que se encuentre en estado de reconciliación de DataLink no posible (DRNP) necesita que se lleve a cabo una acción correctiva (posiblemente fuera de la base de datos). Una vez completada la acción correctiva, la tabla deja el estado DRNP mediante la opción IMMEDIATE UNCHECKED. Deberá utilizarse entonces la API o el mandato RECONCILE para comprobar las restricciones de integridad de DATALINK. Para obtener una referencia más detallada sobre cómo hacer que una tabla deje el estado de reconciliación de DataLink no posible, consulte *Administration Guide*.
  - Mientras se comprueba la integridad se retiene un bloqueo exclusivo en cada tabla especificada en la invocación SET INTEGRITY.
  - Se adquiere un bloqueo compartido en cada tabla que no se lista en la invocación SET INTEGRITY pero es una tabla padre de una de las tablas dependientes que se comprueban.
  - Si se produce un error durante la comprobación de integridad, todos los efectos de la comprobación, incluyendo la supresión en la original e inserción en las tablas de excepciones se retrotraerán.
  - Si la sentencia SET INTEGRITY no es efectiva cuando se emite con una cláusula FORCE GENERATED debido a una falta de espacio para archivos de anotaciones, y este espacio no se puede aumentar lo suficiente, se puede utilizar el mandato **db2gncol** para generar los valores mediante operaciones de confirmación intermitentes. Entonces se puede ejecutar SET INTEGRITY de nuevo, sin la cláusula FORCE GENERATED.

## Ejemplo

*Ejemplo 1:* El siguiente ejemplo es una consulta que nos proporciona información acerca del estado pendiente de comprobación de las tablas. SUBSTR se utiliza para extraer los 2 primeros bytes de la columna CONST\_CHECKED de SYSCAT.TABLES. El primer byte representa las restricciones de clave foránea y el segundo byte representa las restricciones de comprobación.

```
SELECT TABNAME,
 SUBSTR(CONST_CHECKED, 1, 1) AS FK_CHECKED,
 SUBSTR(CONST_CHECKED, 2, 1) AS CC_CHECKED
FROM SYSCAT.TABLES
WHERE STATUS = 'C'
```

## SET INTEGRITY

*Ejemplo 2:* Establezca las tablas T1 y T2 en el estado pendiente de comprobación:

```
SET INTEGRITY FOR T1, T2 OFF
```

*Ejemplo 3:* Compruebe la integridad para T1 y obtenga sólo la primera violación:

```
SET INTEGRITY FOR T1 IMMEDIATE CHECKED
```

*Ejemplo 4:* Compruebe la integridad para T1 y T2 y ponga las filas con violaciones en las tablas de excepciones E1 y E2:

```
SET INTEGRITY FOR T1, T2 IMMEDIATE CHECKED
FOR EXCEPTION IN T1 USE E1,
IN T2 USE E2
```

*Ejemplo 5:* Habilite la comprobación de restricciones FOREIGN KEY en T1 y haga que se evite la comprobación de restricciones CHECK en T2 con la opción IMMEDIATE CHECKED:

```
SET INTEGRITY FOR T1 FOREIGN KEY,
T2 CHECK IMMEDIATE UNCHECKED
```

*Ejemplo 6:* Añada una restricción de comprobación y una clave foránea a la tabla EMP\_ACT, utilizando dos sentencias ALTER TABLE. Para realizar una comprobación de restricciones en una sola pasada de la tabla, la comprobación de la integridad se desactiva antes de las sentencias ALTER y se comprueba después de la ejecución.

```
SET INTEGRITY FOR EMP_ACT OFF;
ALTER TABLE EMP_ACT ADD CHECK (EMSTDATE <= EMENDATE);
ALTER TABLE EMP_ACT ADD FOREIGN KEY (EMPNO) REFERENCES EMPLOYEE;
SET INTEGRITY FOR EMP_ACT IMMEDIATE CHECKED
```

*Ejemplo 7:* Definir la comprobación de integridad para columnas generadas.

```
SET INTEGRITY FOR T1 IMMEDIATE CHECKED
FORCE GENERATED
```



## SET PASSTHRU

La sentencia SET PASSTHRU abre y cierra una sesión para someter directamente el SQL nativo de una fuente de datos a esa fuente de datos. La sentencia no está bajo el control de transacción.

### Invocación

Esta sentencia puede emitirse interactivamente. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben proporcionar autorización para:

- Realizar un paso a través para la fuente de datos.
- Satisfacer las medidas de seguridad en la fuente de datos.

### Sintaxis

```

▶▶—SET PASSTHRU—┬── nombre-servidor ───▶
 └── RESET ───────────▶

```

### Descripción

*nombre-servidor*

Indica la fuente de datos para la que se debe abrir una sesión de paso a través. *nombre-servidor* debe identificar una fuente de datos que se describe en el catálogo.

**RESET**

Cierra una sesión de paso a través.

### Notas

Consulte “Proceso del recurso de paso a través” en la página 1423 para ver las directrices y las restricciones para utilizar el paso a través.

### Ejemplos

*Ejemplo 1:* Inicie una sesión de paso a través para la fuente de datos BACKEND.

```

strcpy (PASS_THRU, "SET PASSTHRU BACKEND");
EXEC SQL EXECUTE IMMEDIATE :PASS_THRU;

```

*Ejemplo 2:* Inicie una sesión de paso a través con una sentencia PREPARE.

```

strcpy (PASS_THRU, "SET PASSTHRU BACKEND");
EXEC SQL PREPARE STMT FROM :PASS_THRU;
EXEC SQL EXECUTE STMT;

```

*Ejemplo 3:* Finalice una sesión de paso a través.

## SET PASSTHRU

```
strcpy (PASS_THRU_RESET, "SET PASSTHRU RESET");
EXEC SQL EXECUTE IMMEDIATE :PASS_THRU_RESET;
```

*Ejemplo 4:* Utilice las sentencias PREPARE y EXECUTE para finalizar una sesión de paso a través.

```
strcpy (PASS_THRU_RESET, "SET PASSTHRU RESET");
EXEC SQL PREPARE STMT FROM :PASS_THRU_RESET;
EXEC SQL EXECUTE STMT;
```

*Ejemplo 5:* Abra una sesión de paso a través para una fuente de datos, cree un índice agrupado para una tabla en esta fuente de datos y cierre la sesión de paso a través.

```
strcpy (PASS_THRU, "SET PASSTHRU BACKEND");
EXEC SQL EXECUTE IMMEDIATE :PASS_THRU;
EXEC SQL PREPARE STMT modalidad de paso a través
FROM "CREATE UNIQUE
 CLUSTERED INDEX TABLE_INDEX
 ON USER2.TABLE la tabla no es un
 WITH IGNORE DUP KEY"; seudónimo
EXEC SQL EXECUTE STMT;
strcpy (PASS_THRU_RESET, "SET PASSTHRU RESET");
EXEC SQL EXECUTE IMMEDIATE :PASS_THRU_RESET;
```

## SET PATH

La sentencia SET PATH cambia el valor del registro especial CURRENT PATH. No está bajo control de transacción.

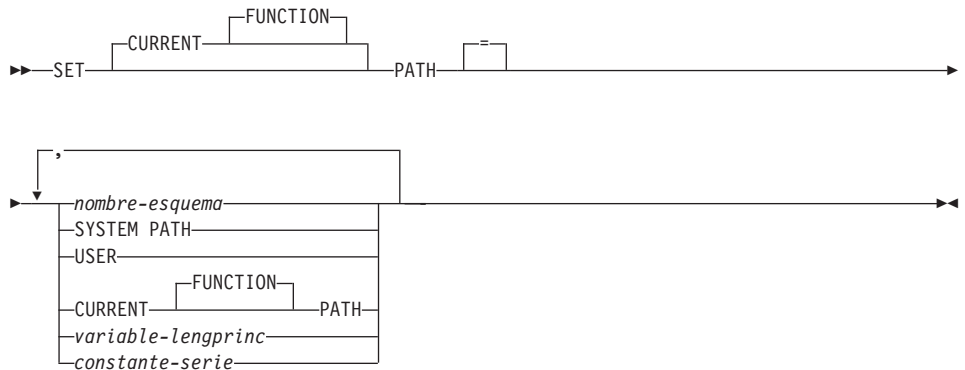
### Invocación

Esta sentencia puede incluirse en un programa de aplicación o emitirse de forma interactiva. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

No se necesita ninguna autorización para ejecutar esta sentencia.

### Sintaxis



### Descripción

#### *nombre-esquema*

Este nombre, que consta de una sola parte, identifica un esquema que existe en el servidor de aplicaciones. No se efectúa ninguna validación de que existe el esquema en el momento en que se establece la vía de acceso. Si, por ejemplo, un *nombre-esquema* está mal escrito, no se capturará y ello podría afectar a las operaciones SQL posteriores.

#### SYSTEM PATH

Este valor es igual a especificar los nombres de esquema "SYSIBM","SYSFUN".

#### USER

El valor del registro especial USER.

#### CURRENT PATH

El valor de CURRENT PATH antes de la ejecución de esta sentencia. También puede especificarse CURRENT FUNCTION PATH.

## SET PATH

### *variable-lengprinc*

Una variable de tipo CHAR o VARCHAR. La longitud del contenido de la *variable-lengprinc* no debe ser mayor que 30 (SQLSTATE 42815). Su valor no puede ser nulo. Si una *variable-lengprinc* tiene asociada una variable indicadora, el valor de dicha variable indicadora no debe indicar un valor nulo (SQLSTATE 42815).

Los caracteres de la *variable-lengprinc* deben estar justificados por la izquierda. Cuando se especifica el *nombre-esquema* con una *variable-lengprinc*, deben especificarse todos los caracteres en mayúsculas o en minúsculas exactamente tal como se desea ya que no se efectúa la conversión a mayúsculas.

### *constante-serie*

Una constante de serie de caracteres con una longitud máxima de 30 bytes.

## Normas

- Un nombre de esquema no puede aparecer más de una vez en la vía de acceso de función (SQLSTATE 42732).
- El número de esquemas que se pueden especificar está limitado por la longitud total del registro especial CURRENT PATH. La serie del registro especial se crea tomando cada nombre de esquema especificado y eliminando los blancos de cola, delimitando con comillas dobles, doblando las comillas dentro del nombre de esquema cuando sea necesario y, después, separando cada nombre de esquema por una coma. La longitud de la serie resultante no puede exceder de 254 bytes (SQLSTATE 42907).

## Notas

- El valor inicial del registro especial CURRENT PATH es "SYSIBM","SYSFUN","X", donde X es el valor del registro especial USER.
- No es necesario especificar el esquema SYSIBM. Si no se incluye en la vía de acceso de SQL, se supone implícitamente que es el primer esquema (en este caso, no se incluye en el registro especial CURRENT PATH).
- El registro especial CURRENT PATH especifica la vía de acceso de SQL utilizada para resolver las funciones, los procedimientos y los tipos de datos definidos por el usuario de las sentencias de SQL dinámicas. La opción de enlace lógico FUNCPATH especifica la vía de acceso de SQL que se ha de utilizar para resolver las funciones y los tipos de datos definidos por el usuario de las sentencias de SQL estáticas. Consulte el manual *Consulta de mandatos* para obtener más información sobre la utilización de la opción FUNCPATH en el mandato BIND.

## Ejemplo

*Ejemplo 1:* La siguiente sentencia establece el registro especial CURRENT FUNCTION PATH.

```
SET PATH = FERMAT, "McDrw #8", SYSIBM
```

*Ejemplo 2:* El ejemplo siguiente recupera el valor actual del registro especial CURRENT PATH en la variable del lenguaje principal denominada CURPATH.

```
EXEC SQL VALUES (CURRENT PATH) INTO :CURPATH;
```

El valor sería "FERMAT","McDrw #8","SYSIBM" si se estableciese por el ejemplo anterior.

## SET SCHEMA

---

### SET SCHEMA

La sentencia SET SCHEMA cambia el valor del registro especial CURRENT SCHEMA. No está bajo control de transacción. Si el paquete está enlazado con la opción DYNAMICRULES BIND, esta sentencia no afecta al calificador utilizado para referencias de objetos de base de datos no calificadas.

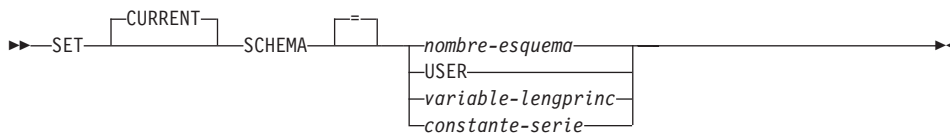
#### Invocación

La sentencia puede incluirse en un programa de aplicación o emitirse interactivamente. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

#### Autorización

No se necesita ninguna autorización para ejecutar esta sentencia.

#### Sintaxis



#### Descripción

##### *nombre-esquema*

Este nombre, que consta de una sola parte, identifica un esquema que existe en el servidor de aplicaciones. La longitud no debe ser mayor que 30 bytes (SQLSTATE 42815). No se efectúa ninguna validación de que existe el esquema en el momento en que se establece el esquema. Si un *nombre-esquema* está escrito incorrectamente, no se capturarán y ello puede afectar a las operaciones subsiguientes de SQL.

##### USER

El valor del registro especial USER.

##### *variable-lengprinc*

Una variable de tipo CHAR o VARCHAR. La longitud del contenido de la *variable-lengprinc* no debe ser mayor que 30 (SQLSTATE 42815). No se puede establecer en nulo. Si una *variable-lengprinc* tiene asociada una variable indicadora, el valor de dicha variable indicadora no debe indicar un valor nulo (SQLSTATE 42815).

Los caracteres de la *variable-lengprinc* deben estar justificados por la izquierda. Cuando se especifica el *nombre-esquema* con una *variable-lengprinc*, deben especificarse todos los caracteres en mayúsculas o en minúsculas exactamente tal como se desea que estén ya que no se realiza ninguna conversión a caracteres en mayúsculas.

*constante-serie*

Es una constante de tipo serie con una longitud máxima de 30.

## Normas

- Si el valor especificado no se adapta a las reglas para un *nombre-esquema*, se produce un error (SQLSTATE 3F000).
- El valor del registro especial CURRENT SCHEMA se utiliza como nombre de esquema en todas las sentencias de SQL dinámicas, con la excepción de la sentencia CREATE SCHEMA, donde existe una referencia no calificada a un objeto de base de datos.
- La opción de enlace lógico QUALIFIER especifica el nombre de esquema a utilizar como calificador para los nombres de objetos de base de datos no calificados en las sentencias de SQL estáticas (consulte la publicación *Consulta de mandatos* para obtener más información sobre la utilización de la opción QUALIFIER).

## Notas

- El valor inicial del registro especial CURRENT SCHEMA equivale a USER.
- El establecimiento del registro especial CURRENT SCHEMA no afecta al registro especial CURRENT PATH. En consecuencia, el registro especial CURRENT SCHEMA no se incluirá en la vía de acceso de SQL y es posible que la resolución de funciones, procedimientos y tipos definidos por el usuario no encuentre estos objetos. Para incluir el valor del esquema actual en la vía de acceso de SQL, siempre que se emita la sentencia SET SCHEMA, emita también la sentencia SET PATH incluyendo el nombre de esquema de la sentencia SET SCHEMA.
- CURRENT SQLID se acepta como sinónimo de CURRENT SCHEMA y el efecto de una sentencia SET CURRENT SQLID será idéntico al de una sentencia SET CURRENT SCHEMA. No tendrán lugar otros efectos, como, por ejemplo, cambios de autorización de la sentencia.

## Ejemplos

*Ejemplo 1:* La sentencia siguiente establece el registro especial CURRENT SCHEMA.

```
SET SCHEMA RICK
```

*Ejemplo 2:* El ejemplo siguiente recupera el valor actual del registro especial CURRENT SCHEMA en la variable del lenguaje principal denominada CURSCHEMA.

```
EXEC SQL VALUES (CURRENT SCHEMA) INTO :CURSCHEMA;
```

El valor sería RICK, establecido en el ejemplo anterior.

## SET SERVER OPTION

---

### SET SERVER OPTION

La sentencia SET SERVER OPTION especifica un valor de opción de servidor que ha de permanecer en vigor mientras un usuario o una aplicación esté conectado a la base de datos federada. Cuando finaliza la conexión, se reinstaura el valor anterior de esta opción de servidor. La sentencia no está bajo el control de transacción.

#### Invocación

Esta sentencia puede emitirse interactivamente. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

#### Autorización

El ID de autorización de la sentencia debe tener la autorización SYSADM o DBADM en la base de datos federada.

#### Sintaxis

```
►► SET SERVER OPTION nombre-opción-servidor TO constante-serie ►►
► FOR SERVER nombre-servidor ►►
```

#### Descripción

*nombre-opción-servidor*

Nombra la opción del servidor que se ha de establecer. Consulte “Opciones de servidor” en la página 1415 para ver las descripciones de las opciones de servidor.

**TO** *constante-serie*

Especifica un valor para *nombre-opción-servidor* como una constante de serie de caracteres. Consulte “Opciones de servidor” en la página 1415 para ver las descripciones de los valores posibles.

**SERVER** *nombre-servidor*

Nombra la fuente de datos a la que se aplica *nombre-opción-servidor*. Debe ser un servidor descrito en el catálogo.

#### Notas

- Los nombres de opción del servidor se pueden entrar en mayúsculas o minúsculas.
- Actualmente SET SERVER OPTION sólo da soporte a la contraseña, fold\_id, y a las opciones de servidor fold\_pw.
- Se pueden someter una o varias sentencias SET SERVER OPTION cuando un usuario o una aplicación se conecta a la base de datos federada. La sentencia (o sentencias) debe especificarse en el inicio de la primera unidad de trabajo que se procese después de establecer la conexión.



## Ejemplos

*Ejemplo 1:* Una base de datos Oracle llamada RATCHIT se define en una base de datos federada llamada DJDB. RATCHIT está configurada para no permitir indicaciones de planes. Sin embargo, el DBA desearía que las indicaciones de planes estuviesen habilitadas para una ejecución de prueba de una aplicación nueva. Cuando la ejecución finalice, las indicaciones de planes se inhabilitarán de nuevo.

```
CONNECT TO DJDB;
strcpy(stmt,"establecer la opción de servidor plan_hints en 'Y' para
 el servidor ratchit");
EXEC SQL EXECUTE IMMEDIATE :stmt;
strcpy(stmt,"seleccionar c1 de ora_t1 donde c1 > 100");
/*Generar indicaciones de planes*/
EXEC SQL PREPARE s1 FROM :stmt;
EXEC SQL DECLARE c1 CURSOR FOR s1;
EXEC SQL OPEN c1;
EXEC SQL FETCH c1 INTO :hv;
```

*Ejemplo 2:* Ha establecido la opción de servidor PASSWORD en 'Y' (validando las contraseñas en la fuente de datos) para todas las fuentes de datos de Oracle 8. Sin embargo, para una sesión en particular en la cual se conecta una aplicación a la base de datos federada para acceder a una fuente de datos Oracle 8 específica —una definida en la base de datos federada DJDB como ORA8A—no será necesario validar las contraseñas.

```
CONNECT TO DJDB;
strcpy(stmt,"establecer la opción de servidor password en 'N' para
 el servidor ora8a");
EXEC SQL PREPARE STMT_NAME FROM :stmt;
EXEC SQL EXECUTE STMT_NAME FROM :stmt;
strcpy(stmt,"seleccionar max(c1) de ora8a_t1");
EXEC SQL PREPARE STMT_NAME FROM :stmt;
EXEC SQL DECLARE c1 CURSOR FOR STMT_NAME;
EXEC SQL OPEN c1; /*No valida la contraseña en ora8a*/
EXEC SQL FETCH c1 INTO :hv;
```

## SET variable

### SET variable

La sentencia SET variable asigna valores a variables locales o a variables de transición nuevas. Está bajo el control de transacción.

#### Invocación

Esta sentencia sólo se puede utilizar como sentencia de SQL en una sentencia dinámica compuesta, un desencadenante, una función SQL o un método SQL.

#### Autorización

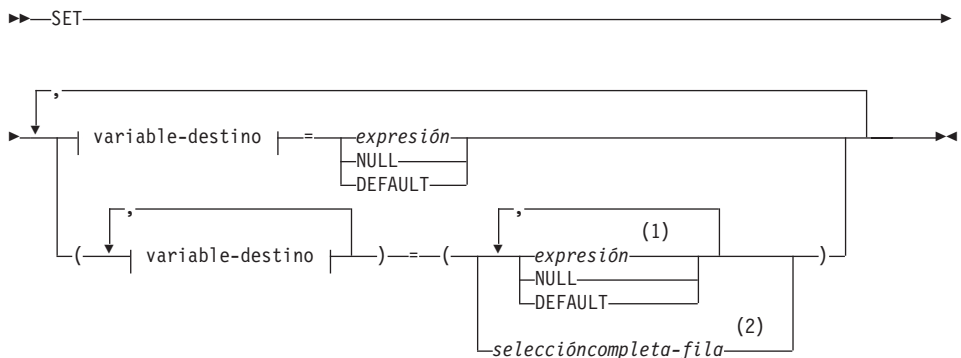
Para hacer referencia a una variable de transición, el ID de autorización del creador de desencadenantes debe tener al menos uno de los privilegios siguientes:

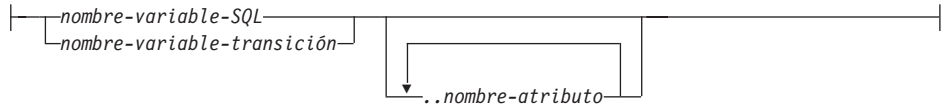
- UPDATE de las columnas a las que se hace referencia en el lado izquierdo de la asignación y SELECT para las columnas a las que se hace referencia en el lado derecho
- Privilegio CONTROL para la tabla (tabla sujeto del desencadenante)
- Autorización SYSADM o DBADM.

Para ejecutar esta sentencia con una *selección-completa-fila* como lado derecho de la asignación, el ID de autorización del definidor de desencadenantes o del propietario de sentencias dinámicas compuestas debe tener también como mínimo uno de los privilegios siguientes para cada tabla o vista a la que se hace referencia:

- Privilegio SELECT
- Privilegio CONTROL
- SYSADM o DBADM.

#### Sintaxis



**variable-destino:****Notas:**

- 1 El número de expresiones, NULL y DEFAULT debe coincidir con el número de *variables-destino*.
- 2 El número de columnas en la lista de selección debe coincidir con el número de *variables-destino*.

**Descripción****variable-destino**

Identifica la variable de destino de la asignación. Una *variable-destino* que represente la misma variable no debe especificarse más de una vez (SQLSTATE 42701).

*nombre-variable-SQL*

Identifica la variable SQL que es el sujeto de la asignación. Las variables SQL se deben declarar antes de utilizarlas. Las variables SQL pueden definirse en una sentencia dinámica compuesta.

*nombre-variable-transición*

Identifica la columna que se debe actualizar en la fila de transición. Un *nombre-variable-transición* debe identificar una columna en la tabla sujeto de un desencadenante, calificado opcionalmente por un nombre de correlación que identifique el nuevo valor (SQLSTATE 42703).

*..nombre de atributo*

Especifica el atributo de un tipo estructurado que está definido (esto se denomina *asignación de atributos*). El *nombre-variable-SQL* o *nombre-variable-transición* especificado debe definirse con un tipo estructurado definido por el usuario (SQLSTATE 428DP). El *nombre-atributo* debe ser un atributo del tipo estructurado (SQLSTATE 42703). Una asignación que no incluye la cláusula *..nombre de atributo* se denomina asignación convencional.

*expresión*

Indica el nuevo valor de la *variable-destino*. La expresión es cualquier expresión del tipo especificado en el apartado "Expresiones" en la página 176. La expresión no puede incluir una función de columna excepto cuando se produce dentro de una selección completa escalar (SQLSTATE 42903). En el contexto de una sentencia CREATE TRIGGER, una *expresión* puede contener referencias a las variables de transición OLD

## SET variable

y NEW y debe estar calificada con el *nombre-correlación* para especificar la variable de transición (SQLSTATE 42702).

### NULL

Especifica el valor nulo y sólo se puede especificar para las columnas que pueden contener nulos (SQLSTATE 23502). NULL no puede ser el valor en una asignación de atributos (SQLSTATE 429B9), a menos que se convirtiera específicamente al tipo de datos del atributo.

### DEFAULT

Especifica que se debe utilizar el valor por omisión.

Si *variable-destino* es una columna, el valor insertado depende de cómo se haya definido la columna en la tabla.

- Si la columna se ha definido utilizando la cláusula WITH DEFAULT, el valor se establece en el valor por omisión definido para la columna (consulte *cláusula-predefinida* en el apartado “ALTER TABLE” en la página 565).
- Si se utiliza la cláusula IDENTITY para definir la columna, el gestor de bases de datos genera el valor.
- Si la columna se ha definido sin especificar las cláusulas WITH DEFAULT, IDENTITY ni NOT NULL, entonces el valor es NULL.
- Si la columna se ha definido utilizando la cláusula NOT NULL y:
  - no se utiliza la cláusula IDENTITY o
  - no se ha utilizado la cláusula WITH DEFAULT o
  - se ha utilizado la cláusula DEFAULT NULL

la clave DEFAULT no se puede especificar para dicha columna (SQLSTATE 23502).

Si *variable-destino* es una variable SQL, el valor insertado es el valor por omisión especificado o implícito en la declaración de variable.

### *selección-completa-fila*

Una selección completa que devuelve una sola fila con el número de columnas correspondiente al número de variables-destino especificadas para la asignación. Los valores se asignan a cada variable-destino correspondiente. Si el resultado de la selección-completa-fila es ninguna fila, se asignan los valores nulos. En el contexto de una sentencia CREATE TRIGGER, una *selección-completa-fila* puede contener referencias a las variables de transición OLD y NEW que deben estar calificadas con su *nombre-correlación* para especificar qué variable de transición se debe utilizar (SQLSTATE 42702). Se devuelve un error si hay más de una fila en el resultado (SQLSTATE 21000).

## Normas

- El número de valores que se deben asignar desde las expresiones, NULL y DEFAULT o la *selección-completa-fila* debe coincidir con el número de variables-destino especificadas para la asignación (SQLSTATE 42802).
- Una sentencia SET variable no puede asignar una variable SQL y una variable de transición en una sentencia (SQLSTATE 42997).
- Los valores se asignan a las variables-destino bajo las reglas de asignación descritas en el apartado “Asignaciones y comparaciones” en la página 105.
- Si la sentencia se utiliza en un desencadenante BEFORE UPDATE y la variable de registro DB2\_UPDATE\_PART\_KEY=OFF (el valor por omisión), una *variable-transición* especificada como *variable-destino* no puede ser una columna clave de particionamiento (SQLSTATE 42997).

## Notas

- Si se incluye más de una asignación, todas las *expresiones y selecciones-completas-fila* se evalúan antes de que se realicen las asignaciones. Por lo tanto, las referencias a variables-destino en una expresión o una selección completa de fila son siempre el valor de la variable-destino antes de cualquier asignación en la única sentencia SET.
- Cuando se actualiza una columna de identidad definida como un tipo diferenciado, el cálculo completo se realiza en el tipo fuente y el resultado se convierte al tipo diferenciado antes de que se asigne realmente el valor a la columna.<sup>107</sup>
- Para que DB2 genere un valor en una sentencia SET para una columna de identidad, utilice la palabra clave DEFAULT:

```
SET NEW.EMPNO = DEFAULT
```

En este ejemplo, NEW.EMPNO está definido como columna de identidad, y el valor utilizado para actualizar la columna es generado por DB2.

- Consulte el apartado “INSERT” en la página 1082 para obtener más información sobre el consumo de valores de una secuencia generada para una columna de identidad y para obtener información sobre cómo se excede el valor máximo de una columna de identidad.

## Ejemplos

*Ejemplo 1:* Establezca la columna de salario de la fila para la que se está ejecutando actualmente la acción del desencadenante en 50000.

```
SET NEW_VAR.SALARY = 50000;
0
SET (NEW_VAR.SALARY) = (50000);
```

---

107. No hay ninguna conversión del valor anterior al tipo fuente antes del cálculo.

## SET variable

*Ejemplo 2:* Establezca la columna de salario y de comisión de la fila para la que se está ejecutando actualmente la acción del desencadenante en 50000 y 8000 respectivamente.

```
SET NEW_VAR.SALARY = 50000, NEW_VAR.COMM = 8000;
o
SET (NEW_VAR.SALARY, NEW_VAR.COMM) = (50000, 8000);
```

*Ejemplo 3:* Establezca la columna de salario y de comisión de la fila para la que se está ejecutando actualmente la acción del desencadenante en el promedio del salario y de la comisión de los empleados del departamento de la fila actualizada respectivamente.

```
SET (NEW_VAR.SALARY, NEW_VAR.COMM)
= (SELECT AVG(SALARY), AVG(COMM)
FROM EMPLOYEE E
WHERE E.WORKDEPT = NEW_VAR.WORKDEPT);
```

*Ejemplo 4:* Establezca la columna de salario y de comisión de la fila para la que se está ejecutando actualmente la acción del desencadenante en 10000 y el valor original del salario respectivamente (p. ej., antes de que se ejecutarse la sentencia SET).

```
SET NEW_VAR.SALARY = 10000, NEW_VAR.COMM = NEW_VAR.SALARY;
o
SET (NEW_VAR.SALARY, NEW_VAR.COMM) = (10000, NEW_VAR.SALARY);
```

---

## UPDATE

La sentencia UPDATE actualiza los valores de las columnas especificadas en las filas de una tabla o vista. La actualización de una fila de una vista actualiza una fila de su tabla base.

Las formas de esta sentencia son:

- La forma UPDATE *Con búsqueda* se utiliza para actualizar una o varias filas (determinadas opcionalmente por la condición de búsqueda).
- La forma de UPDATE *Con posición* se utiliza para actualizar exactamente una fila (tal como determina la posición actual de un cursor).

### Invocación

Una sentencia UPDATE puede incluirse en un programa de aplicación o emitirse mediante la utilización de sentencias de SQL dinámicas. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Privilegio UPDATE para la tabla o vista cuyas filas se deben actualizar
- Privilegio UPDATE para cada una de las columnas que se deben actualizar.
- Privilegio CONTROL para la tabla o vista cuyas filas se deben actualizar
- Autorización SYSADM o DBADM.
- Si se incluye una *selección completa-fila* en la asignación, como mínimo uno de los siguientes para cada tabla o vista referenciada:
  - Privilegio SELECT
  - Privilegio CONTROL
  - Autorización SYSADM o DBADM.

Para cada tabla o vista referenciada por una subconsulta, el ID de autorización de la sentencia debe tener también como mínimo uno de los privilegios siguientes:

- Privilegio SELECT
- Privilegio CONTROL
- Autorización SYSADM o DBADM.

Cuando se precompila el paquete con reglas SQL92 <sup>108</sup> y la forma con búsqueda de UPDATE incluye una referencia a una columna de la tabla o vista del lado derecho de la *cláusula-asignación* o en cualquier sitio de la

---

108. El paquete que se utiliza para procesar la sentencia se precompila utilizando la opción LANGLEVEL con el valor SQL92E o MIA.

# UPDATE

*condición-búsqueda*, los privilegios que tiene el ID de autorización de la sentencia también deben incluir como mínimo uno de los siguientes:

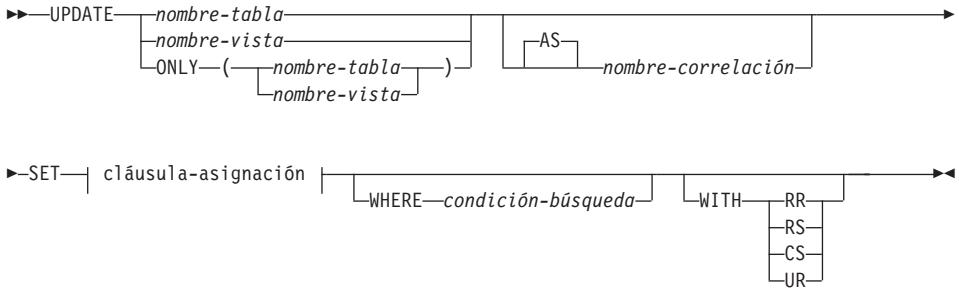
- Privilegio SELECT
- Privilegio CONTROL
- Autorización SYSADM o DBADM.

Cuando la vista o tabla especificada está precedida de la palabra clave ONLY, los privilegios que tiene el ID de autorización de la sentencia también deben incluir el privilegio SELECT para cada subvista o subtabla de la vista o tabla especificada.

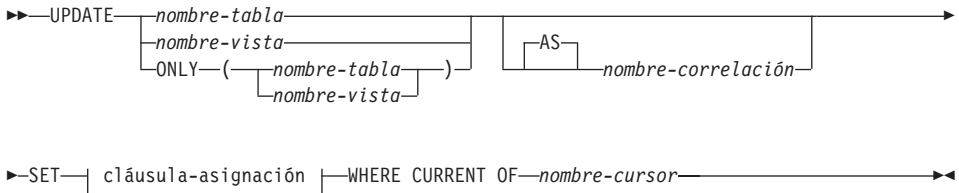
No se comprueban los privilegios GROUP para las sentencias UPDATE estáticas.

## Sintaxis

### UPDATE con búsqueda:

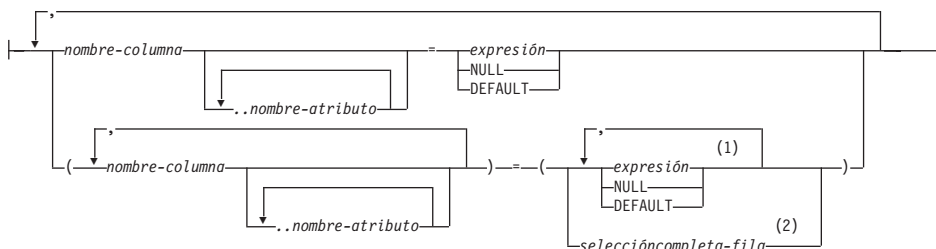


### UPDATE con posición:



### cláusula-asignación:



**Notas:**

- 1 El número de expresiones, de NULL y de DEFAULT debe coincidir con el número de *nombres-columna*.
- 2 El número de columnas en la lista de selección debe coincidir con el número de *nombres-columna*.

**Descripción***nombre-tabla* o *nombre-vista*

Es el nombre de la tabla o vista que se ha de actualizar. El nombre debe identificar una tabla o vista descrita en el catálogo, pero no una tabla de catálogo, una vista de una tabla de catálogo (a menos que sea una vista SYSSTAT actualizable), una tabla de resumen, una vista de sólo lectura ni un apodo. (Para ver una explicación de las vistas de sólo lectura, consulte el apartado "CREATE VIEW" en la página 957. Para ver una explicación de las vistas de catálogo actualizables, consulte el "Apéndice D. Vistas de catálogo" en la página 1281.)

Si *nombre-tabla* es una tabla con tipo, la sentencia puede actualizar filas de la tabla o cualquiera de sus subtablas correspondientes. Sólo pueden establecerse o referirse las columnas de la tabla especificada en la cláusula WHERE. Para una sentencia UPDATE con posición, el cursor asociado también debe haber especificado la misma tabla o vista en la cláusula FROM sin utilizar ONLY.

**ONLY** (*nombre-tabla*)

Aplicable a las tablas con tipo, la palabra clave ONLY especifica que la sentencia sólo se debe aplicar a los datos de la tabla especificada y no puede actualizar las filas de las subtablas correspondientes. Para una sentencia UPDATE con posición, el cursor asociado también debe haber especificado la tabla en la cláusula FROM utilizando ONLY. Si *nombre-tabla* no es una tabla con tipo, la palabra clave ONLY no tiene efecto en la sentencia.

**ONLY** (*nombre-vista*)

Aplicable a las vistas con tipo, la palabra clave ONLY especifica que la sentencia sólo se debe aplicar a los datos de la vista especificada y no puede actualizar las filas de las subvistas correspondientes. Para una

## UPDATE

sentencia UPDATE con posición, el cursor asociado también debe haber especificado la vista en la cláusula FROM utilizando ONLY. Si *nombre-vista* no es una vista con tipo, la palabra clave ONLY no tiene efecto en la sentencia.

### AS

Palabra clave opcional para introducir el *nombre-correlación*.

#### *nombre-correlación*

Se puede utilizar en *condición-búsqueda* o *cláusula-asignación* para designar la tabla o vista. Para obtener una explicación de *nombre-correlación*, consulte el apartado “Nombres de correlación” en la página 144.

### SET

Introduce la asignación de valores a nombres de columna.

#### *cláusula-asignación*

##### *nombre-columna*

Identifica una columna que se ha de actualizar. El *nombre-columna* debe identificar una columna actualizable de la vista o tabla especificada.<sup>109</sup> La columna de ID de objeto de una tabla con tipo no es actualizable (SQLSTATE 428DZ). Una columna no debe especificarse más de una vez, a no ser que vaya seguida de *..nombre-atributo* (SQLSTATE 42701).

Para una UPDATE con posición:

- Si se ha especificado la cláusula UPDATE en la *sentencia-select* del cursor, cada nombre de columna de la *cláusula-asignación* debe aparecer también en la cláusula UPDATE.
- Si la cláusula UPDATE no se ha especificado en la *sentencia-select* del cursor y se ha especificado LANGLEVEL MIA o SQL92E al precompilar la aplicación, se puede especificar el nombre de cualquier columna actualizable.
- Si la cláusula UPDATE no se ha especificado en la *sentencia-select* del cursor y se ha especificado LANGLEVEL SAA1 explícitamente o por omisión al precompilar la aplicación, no se puede actualizar ninguna columna.

##### *.. nombre-atributo*

Especifica el atributo de un tipo estructurado que está definido (esto se denomina *asignación de atributos*). El *nombre-columna* especificado se debe definir con un tipo estructurado definido por el usuario (SQLSTATE 428DP). El nombre de atributo debe ser un atributo del tipo estructurado del *nombre-columna* (SQLSTATE 42703). Las

---

109. Una columna de una clave de particionamiento es actualizable, a no ser que el parámetro de configuración DB2\_UPDATE\_PART\_KEY esté establecido en 'OFF' (SQLSTATE 42997). Debe suprimirse o insertarse la fila de datos para cambiar las columnas de una clave de particionamiento.

asignaciones en las que no interviene la cláusula *..nombre-atributo* se denominan *asignaciones convencionales*.

### *expresión*

Indica el nuevo valor de la columna. La expresión es cualquier expresión del tipo especificado en el apartado “Expresiones” en la página 176. La expresión no puede incluir ninguna función de columna a excepción de cuando se produce dentro de una selección completa escalar (SQLSTATE 42903).

Una *expresión* puede contener referencias a las columnas de la tabla de destino de la sentencia UPDATE. Para cada fila que se actualiza, el valor de dicha columna en una expresión es el valor de la columna en la fila antes de que se actualice la fila.

### NULL

Especifica el valor nulo y sólo se puede especificar para las columnas que pueden contener nulos (SQLSTATE 23502). NULL no puede ser el valor en una asignación de atributos (SQLSTATE 429B9), a menos que se convirtiera específicamente al tipo de datos del atributo.

### DEFAULT

Especifica que debe utilizarse el valor por omisión basándose en cómo se define la columna correspondiente en la tabla. El valor que se inserta depende de cómo se ha definido la columna.

- Si la columna se definió como columna generada basándose en una expresión, el sistema genera el valor de la columna de acuerdo con esa expresión.
- Si se utiliza la cláusula IDENTITY para definir la columna, el gestor de bases de datos genera el valor.
- Si la columna se ha definido utilizando la cláusula WITH DEFAULT, el valor se establece en el valor por omisión definido para la columna (consulte la cláusula-predefinida en el apartado “ALTER TABLE” en la página 565).
- Si la columna se ha definido sin especificar las cláusulas WITH DEFAULT, GENERATED ni NOT NULL, entonces el valor utilizado es NULL.
- Si para definir la columna se utiliza la cláusula NOT NULL y no la cláusula GENERATED, o no se utiliza WITH DEFAULT o se utiliza DEFAULT NULL, no se puede especificar la palabra clave DEFAULT para esa columna (SQLSTATE 23502).

El único valor que se puede insertar en una columna generada que se ha definido con la cláusula GENERATED ALWAYS es DEFAULT (SQLSTATE 428C9).

## UPDATE

La palabra clave DEFAULT no se puede utilizar como valor en una asignación de atributos (SQLSTATE 429B9).

### *seleccióncompleta-fila*

Una selección completa que devuelve una sola fila con el número de columnas correspondiente al número de *nombres-columna* especificados para la asignación. Los valores se asignan a cada *nombre-columna* correspondiente. Si el resultado de la *seleccióncompleta-fila* es ninguna fila, se asignan los valores nulos.

Una *seleccióncompleta-fila* puede contener referencias a las columnas de la tabla de destino de la sentencia UPDATE. Para cada fila que se actualiza, el valor de dicha columna en una expresión es el valor de la columna en la fila antes de que se actualice la fila. Se devuelve un error si hay más de una fila en el resultado (SQLSTATE 21000).

## WHERE

Introduce una condición que indica qué filas se actualizan. Puede omitir la cláusula, proporcionar una condición de búsqueda o nombrar un cursor. Si se omite la cláusula, se actualizan todas las filas de la tabla o de la vista.

### *condición-búsqueda*

Es cualquier condición de búsqueda, tal como se describe en el “Capítulo 3. Elementos del lenguaje” en la página 69. Cada *nombre-columna* de la condición de búsqueda, que no sea en una subconsulta, debe nombrar una columna de la tabla o vista. Cuando la condición de búsqueda incluye una subconsulta en la que la misma tabla es el objeto base de UPDATE y de la subconsulta, la subconsulta se evalúa completamente antes de que se actualice cualquier fila.

La condición-búsqueda se aplica a cada fila de la tabla o vista y las filas actualizadas son aquellas para las que el resultado de la condición-búsqueda es verdadero.

Si la condición de búsqueda contiene una subconsulta, la subconsulta puede considerarse como ejecutada cada vez que la condición de búsqueda se aplica a una fila y el resultado se utiliza en la aplicación de la condición de búsqueda. En realidad, una subconsulta sin referencias correlacionadas se ejecuta una sola vez, mientras que una subconsulta con una referencia correlacionada puede tener que ejecutarse una vez para cada fila.

### **CURRENT OF** *nombre-cursor*

Identifica el cursor que se ha de utilizar en la operación de actualización. El *nombre-cursor* debe identificar un cursor declarado tal como se explica en el apartado “DECLARE CURSOR” en la página 976. La sentencia DECLARE CURSOR debe preceder a la sentencia UPDATE en el programa.

La tabla o vista nombrados también deben nombrarse en la cláusula FROM de la sentencia SELECT del cursor, y la tabla resultante del cursor no debe ser de sólo lectura. (Para ver una explicación de las tablas resultantes de sólo lectura, consulte el apartado “DECLARE CURSOR” en la página 976.)

Cuando se ejecuta la sentencia UPDATE, el cursor debe posicionarse en una fila; dicha fila se actualiza.

Esta modalidad de UPDATE no se puede utilizar si el objeto actualizado es una vista que contiene una función OLAP en la lista de selección de la selección completa definidora de la vista (SQLSTATE 42828).

### WITH

Especifica el nivel de aislamiento en el que se ejecuta la sentencia UPDATE.

#### RR

Lectura repetible

#### RS

Estabilidad de lectura

#### CS

Estabilidad del cursor

#### UR

Lectura no confirmada

El nivel de aislamiento por omisión de la sentencia es el nivel de aislamiento del paquete en el que está enlazada la sentencia.

## Normas

- **Asignación:** Los valores de actualización se asignan a las columnas bajo las reglas de asignación descritas en el Capítulo 3.
- **Validez:** La fila actualizada debe ajustarse a cualquier restricción impuesta en la tabla (o en la tabla base de la vista) por el índice de unicidad en una columna actualizada.

Si se utiliza una vista que no se ha definido utilizando WITH CHECK OPTION, se pueden cambiar las filas para que no se ajusten más a la definición de la vista. Dichas filas se actualizan en la tabla base de la vista y ya no aparecen más en la vista.

Si se utiliza una vista que se ha definido utilizando WITH CHECK OPTION, una fila actualizada debe ajustarse a la definición de la vista. Para ver una explicación de las reglas que rigen esta situación, consulte el apartado “CREATE VIEW” en la página 957.

## UPDATE

- **Restricción de comprobación:** El valor de actualización debe cumplir las condiciones-control de las restricciones de comprobación definidas en la tabla.

En UPDATE para una tabla con restricciones de comprobación definidas se evalúan las condiciones de restricción para cada columna una vez para cada fila que se actualiza. Cuando se procesa una sentencia UPDATE, sólo se comprueban las restricciones de comprobación que hacen referencia a las columnas actualizadas.

- **Integridad de referencia:** El valor de las claves de unicidad padre no puede cambiarse si la regla de actualización es RESTRICT y hay una o más filas dependientes. Sin embargo, si la regla de actualización es NO ACTION, las claves de unicidad padres pueden actualizarse siempre que cada hijo tenga una clave padre en el momento en que se completa la sentencia de actualización. Un valor de actualización no nulo para una clave foránea debe ser igual a un valor de la clave primaria de la tabla padre de la relación.

### Notas

- Si un valor de actualización viola cualquier restricción o si se produce cualquier otro error durante la ejecución de la sentencia UPDATE, no se actualiza ninguna fila. El orden en que se actualizan múltiples filas no está definido.
- Cuando una sentencia UPDATE completa su ejecución, el valor de SQLERRD(3) en la SQLCA es el número de filas actualizadas. El campo SQLERRD(5) contiene el número de filas insertadas, suprimidas o actualizadas por todos los desencadenantes activados. Para ver una descripción de la SQLCA, consulte el "Apéndice B. Comunicaciones SQL (SQLCA)" en la página 1261.
- A menos que ya existan los bloqueos adecuados, se adquieren uno o varios bloqueos por la ejecución de una sentencia UPDATE satisfactoria. Hasta que se liberan los bloqueos, la fila actualizada sólo puede accederse por el proceso de aplicación que ha realizado la actualización (excepto las aplicaciones que utilizan el nivel de aislamiento de Lectura no confirmada). Para obtener más información sobre el bloqueo, consulte las descripciones de las sentencias COMMIT, ROLLBACK y LOCK TABLE.
- Si se actualiza el valor del URL de una columna DATALINK, es lo mismo que suprimir el valor DATALINK antiguo e insertar el nuevo. En primer lugar, si el valor antiguo estaba enlazado con un archivo, este archivo se desenlaza. A continuación, a no ser que los atributos de enlace del valor DATALINK estén vacíos, el archivo especificado se enlaza con esta columna.

El valor del comentario de una columna DATALINK puede actualizarse sin volver a enlazar el archivo si se especifica una serie vacía como vía de

acceso de URL (por ejemplo, como argumento *ubicación-datos* de la función escalar DLVALUE o si se especifica el valor nuevo de manera que sea el mismo que el valor antiguo).

Si una columna DATALINK se actualiza con un valor nulo, es lo mismo que suprimir el valor DATALINK existente.

Puede producirse un error cuando se intente actualizar un valor DATALINK si el servidor de archivos del valor existente o del valor nuevo ya no está registrado con el servidor de bases de datos (SQLSTATE 55022).

- Cuando se actualicen las estadísticas de distribución de columna para una tabla con tipo, debe especificarse la subtabla que haya introducido primero la columna.
- Puede haber varias asignaciones de atributos en la misma columna de tipo estructurado, en el orden especificado por la cláusula SET.
- La asignación de atributos invoca el método mutador para el atributo del tipo estructurado definido por el usuario. Por ejemplo, la asignación `st..a1=x` tiene el mismo efecto que utilizar el método mutador en la asignación `st = st..a1(x)`.
- Aunque una columna determinada sólo puede ser objeto de una sola asignación convencional, puede intervenir en varias asignaciones de atributos (pero únicamente si no es objeto de una asignación convencional).
- Cuando se actualiza una columna de identidad definida como tipo diferenciado, el cálculo completo se realiza en el tipo fuente, y el resultado se convierte al tipo diferenciado antes de asignar realmente el valor a la columna.<sup>110</sup>
- Para hacer que DB2 genere un valor en una sentencia SET para una columna de identidad, utilice la palabra clave DEFAULT:

```
SET NEW.EMPNO = DEFAULT
```

En este ejemplo, NEW.EMPNO está definido como columna de identidad, y el valor utilizado para actualizar la columna es generado por DB2.

- Vea "INSERT" en la página 1082 para obtener más información sobre el consumo de valores de una secuencia generada para una columna de identidad.
- Vea "INSERT" en la página 1082 para obtener más información sobre el rebasamiento del valor máximo para una columna de identidad.

## Ejemplos

- *Ejemplo 1:* Cambie el trabajo (JOB) del empleado número (EMPNO) '000290' en la tabla EMPLOYEE por 'LABORER'.

---

110. Antes del cálculo no se realiza ninguna conversión del valor anterior al tipo fuente.

## UPDATE

```
UPDATE EMPLOYEE
SET JOB = 'LABORER'
WHERE EMPNO = '000290'
```

- *Ejemplo 2:* Aumente las personas que trabajan en el proyecto (PRSTAFF) en 1,5 para todos los proyectos de los cuales sea responsable el departamento (DEPTNO) 'D21' en la tabla PROJECT.

```
UPDATE PROJECT
SET PRSTAFF = PRSTAFF + 1,5
WHERE DEPTNO = 'D21'
```

- *Ejemplo 3:* Todos los empleados excepto el director del departamento (WORKDEPT) 'E21' se han vuelto a asignar temporalmente. Indique esto cambiando su trabajo (JOB) por NULL y los valores de su paga (SALARY, BONUS, COMM) a cero en la tabla EMPLOYEE.

```
UPDATE EMPLOYEE
SET JOB=NULL, SALARY=0, BONUS=0, COMM=0
WHERE WORKDEPT = 'E21' AND JOB <> 'MANAGER'
```

Esta sentencia también se podría escribir de la siguiente manera.

```
UPDATE EMPLOYEE
SET (JOB, SALARY, BONUS, COMM) = (NULL, 0, 0, 0)
WHERE WORKDEPT = 'E21' AND JOB <> 'MANAGER'
```

- *Ejemplo 4:* Actualice la columna del salario y comisión del empleado con el número de empleado 000120 por el promedio del salario y la comisión de los empleados del departamento de la fila actualizada respectivamente.

```
UPDATE EMPLOYEE EU
SET (EU.SALARY, EU.COMM)
=
(SELECT AVG(ES.SALARY), AVG(ES.COMM)
FROM EMPLOYEE ES
WHERE ES.WORKDEPT = EU.WORKDEPT)
WHERE EU.EMPNO = '000120'
```

- *Ejemplo 5:* En un programa C visualice las filas de la tabla EMPLOYEE y, después, si se le pide hacerlo, cambie el trabajo (JOB) de algunos empleados por el nuevo trabajo escrito.

```
EXEC SQL DECLARE C1 CURSOR FOR
SELECT *
FROM EMPLOYEE
FOR UPDATE OF JOB;

EXEC SQL OPEN C1;

EXEC SQL FETCH C1 INTO ... ;

if (strcmp (change, "YES") == 0)
EXEC SQL UPDATE EMPLOYEE
SET JOB = :newjob
WHERE CURRENT OF C1;

EXEC SQL CLOSE C1;
```



- *Ejemplo 6:* Mutación de atributos de objetos de columnas.

Sean los siguientes tipos y tablas:

```

CREATE TYPE POINT AS (X INTEGER, Y INTEGER)
NOT FINAL WITHOUT COMPARISONS
MODE DB2SQL

CREATE TYPE CIRCLE AS (RADIUS INTEGER, CENTER POINT)
NOT FINAL WITHOUT COMPARISONS
MODE DB2SQL

CREATE TABLE CIRCLES (ID INTEGER, OWNER VARCHAR(50), C CIRCLE

```

El ejemplo siguiente actualiza la tabla CIRCLES cambiando la columna OWNER y el atributo RADIUS de la columna CIRCLE cuyo ID es 999:

```

UPDATE CIRCLES
SET OWNER = 'Bruce'
 C..RADIUS = 5
WHERE ID = 999

```

El ejemplo siguiente traslada las coordenadas X e Y del centro del círculo identificado por 999:

```

UPDATE CIRCLES
SET C..CENTER..X = C..CENTER..Y,
 C..CENTER..Y = C..CENTER..X
WHERE ID = 999

```

El ejemplo siguiente constituye otro modo de escribir las dos sentencias anteriores. Este ejemplo combina los efectos de ambos ejemplos anteriores:

```

UPDATE CIRCLES
SET (OWNER,C..RADIUS,C..CENTER..X,C..CENTER..Y) =
 ('Bruce',5,C..CENTER..Y,C..CENTER..X)
WHERE ID = 999

```

## VALUES

---

### VALUES

La sentencia VALUES es una forma de consulta. Puede incluirse en un programa de aplicación o emitirse interactivamente. Para ver información detallada, consulte el apartado “selección completa” en la página 512.

## VALUES INTO

La sentencia VALUES INTO produce una tabla resultante que consta de como máximo una fila y asigna los valores de dicha fila a las variables del lenguaje principal.

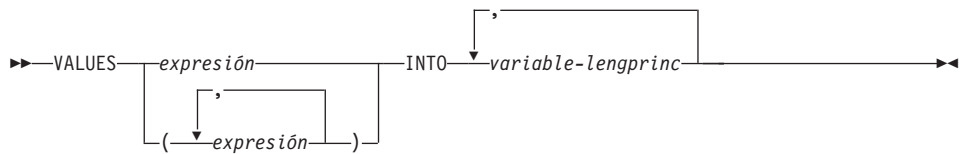
### Invocación

Esta sentencia sólo se puede incluir en un programa de aplicación. Se trata de una sentencia ejecutable que no puede prepararse de forma dinámica.

### Autorización

No se necesita.

### Sintaxis



### Descripción

#### VALUES

Introduce una sola fila que consta de una o varias columnas.

##### *expresión*

Una expresión que define un solo valor de una tabla resultante de una columna. No se soportan estructuras de sistema principal.

##### *(expresión,...)*

Una o más expresiones que definen los valores para una o varias columnas de la tabla resultante. No se soportan estructuras de sistema principal.

#### INTO

Introduce una lista de variables del lenguaje principal.

##### *variable-lengprinc*

Identifica una variable que está descrita en el programa bajo las reglas para la declaración de variables del lenguaje principal.

El primer valor de la fila del resultado se asigna a la primera variable de la lista, el segundo valor a la segunda variable, etcétera. Si el número de variables del lenguaje principal es menor que el número de valores de columna, se asigna el valor 'W' al campo SQLWARN3 de la SQLCA. (Consulte el "Apéndice B. Comunicaciones SQL (SQLCA)" en la página 1261.)

## VALUES INTO

Cada asignación a una variable se realiza de acuerdo a las reglas descritas en el “Asignaciones y comparaciones” en la página 105. Las asignaciones se realizan en secuencia en la lista.

Si se produce un error, no se asigna ningún valor a la variable del lenguaje principal.

### Ejemplos

*Ejemplo 1:* Este ejemplo C recupera el valor del registro especial CURRENT PATH en una variable del lenguaje principal.

```
EXEC SQL VALUES(CURRENT PATH)
INTO :hv1;
```

*Ejemplo 2:* Este ejemplo C recupera una parte de un campo LOB en una variable del lenguaje principal, utilizando el localizador de LOB para la recuperación diferida.

```
EXEC SQL VALUES (substr(:locator1,35))
INTO :details;
```

## WHENEVER

La sentencia **WHENEVER** especifica la acción que se ha de tomar cuando se produce una condición de excepción especificada.

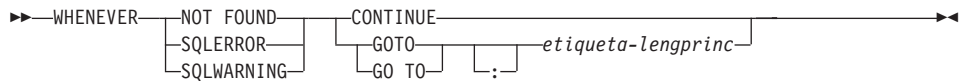
### Invocación

Esta sentencia sólo puede incluirse en un programa de aplicación. No es una sentencia ejecutable. La sentencia no está soportada en REXX.

### Autorización

No se necesita.

### Sintaxis



### Descripción

Las cláusulas **NOT FOUND**, **SQLERROR** o **SQLWARNING** se utilizan para identificar el tipo de condición de excepción.

#### NOT FOUND

Identifica cualquier condición que dé como resultado un **SQLCODE** de +100 o un **SQLSTATE** de '02000'.

#### SQLERROR

Identifica cualquier condición que dé como resultado un **SQLCODE** negativo.

#### SQLWARNING

Identifica cualquier condición que dé como resultado una condición de aviso (**SQLWARN0** es 'W'), o que dé como resultado un código de retorno **SQL** positivo que no sea +100.

Las cláusulas **CONTINUE** o **GO TO** se utilizan para especificar lo que ocurre cuando existe el tipo de condición de excepción identificado.

#### CONTINUE

Provoca que se ejecute la siguiente instrucción secuencial del programa fuente.

#### GOTO o GO TO *etiqueta-lengprinc*

Provoca que el control pase a la sentencia identificada por la *etiqueta-lengprinc*. Para *etiqueta-lengprinc*, utilice un símbolo individual, precedido opcionalmente por dos puntos. El formato del símbolo depende del lenguaje principal.

## WHENEVER

### Notas

Hay tres tipos de sentencias **WHENEVER**:

- **WHENEVER NOT FOUND**
- **WHENEVER SQLERROR**
- **WHENEVER SQLWARNING**

Cada sentencia de SQL ejecutable en un programa está dentro del ámbito de una sentencia **WHENEVER** implícita o explícita de cada tipo. El ámbito de una sentencia **WHENEVER** está relacionado con la secuencia de listado de las sentencias del programa, no con su secuencia de ejecución.

Una sentencia de SQL está dentro del ámbito de la última sentencia **WHENEVER** de cada tipo que se especifica antes de la sentencia de SQL en el programa fuente. Si una sentencia **WHENEVER** de cualquier tipo no se especifica antes de una sentencia de SQL, dicha sentencia de SQL está dentro del ámbito de una sentencia **WHENEVER** de dicho tipo en la que se especifica **CONTINUE**.

### Ejemplo

En el ejemplo C siguiente, si se produce un error, vaya a **HANDLERR**. Si se produce un código de aviso, continúe con el flujo normal del programa. Si no se devuelven datos, vaya a **ENDDATA**.

```
EXEC SQL WHENEVER SQLERROR GOTO HANDLERR;
EXEC SQL WHENEVER SQLWARNING CONTINUE;
EXEC SQL WHENEVER NOT FOUND GO TO ENDDATA;
```

---

## Capítulo 7. Sentencias de control de SQL

Las sentencias de control son sentencias de SQL que permiten utilizar el lenguaje de consulta estructurada de un modo similar al utilizando en la escritura de un programa en un lenguaje de programación estructurado. Las sentencias de control de SQL pueden utilizarse en el cuerpo de una rutina, un desencadenante o una sentencia dinámica compuesta.

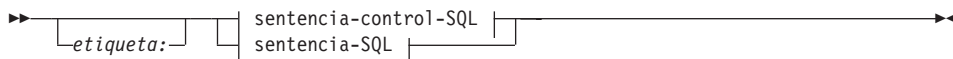
Este capítulo contiene diagramas de sintaxis, descripciones semánticas, reglas y ejemplos de utilización de las sentencias que constituyen el cuerpo del procedimiento de una rutina, un desencadenante o una sentencia dinámica compuesta de SQL.

## Sentencia de procedimiento SQL

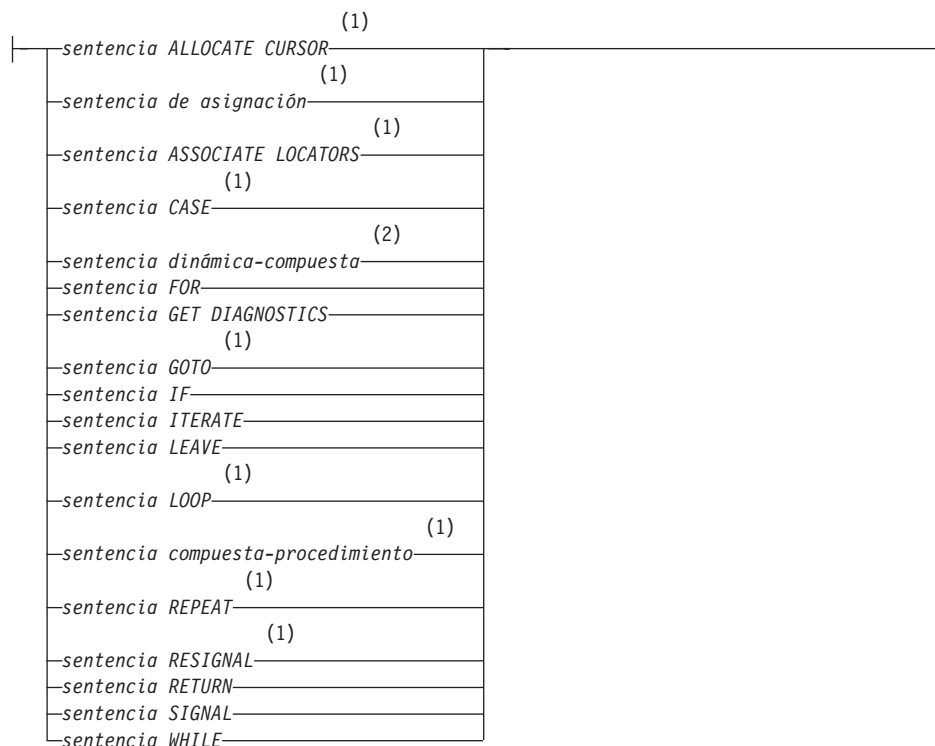
### Sentencia de procedimiento SQL

Una sentencia de procedimiento SQL es una sentencia de control de SQL o una sentencia de SQL que se especifica en una sentencia de control de un cuerpo de rutina SQL.

#### Sintaxis



#### sentencia-control-SQL:



#### Notas:

- 1 Esta sentencia sólo se soporta en el ámbito de un procedimiento SQL.
- 2 Esta sentencia sólo se soporta en un desencadenante, una función SQL o un método SQL. Tiene que ser la sentencia situada en último lugar.



### Descripción

*etiqueta:*

Especifica la etiqueta de una sentencia de procedimiento SQL. La etiqueta debe ser exclusiva dentro de una lista de sentencias de procedimiento SQL, incluidas las sentencias compuestas anidadas dentro de la lista. Observe que las sentencias compuestas que no están anidadas pueden utilizar la misma etiqueta. Varias sentencias de control SQL admiten la especificación de una lista de sentencias de procedimiento SQL.

En el contexto de un desencadenante, una función o un método SQL o una sentencia dinámica compuesta, sólo se pueden etiquetar la sentencia dinámica compuesta, la sentencia FOR y la sentencia WHILE.

### sentencia-SQL

En el cuerpo de un procedimiento SQL, pueden estar contenidas todas las sentencias de SQL ejecutables, excepto las siguientes:

- CONNECT
- CREATE, para cualquier objeto excepto índices, tablas o vistas
- DESCRIBE
- DISCONNECT
- DROP, para cualquier objeto excepto índices, tablas o vistas
- FLUSH EVENT MONITOR
- REFRESH TABLE
- RELEASE (sólo conexión)
- RENAME TABLE
- RENAME TABLESPACE
- REVOKE
- SET CONNECTION
- SET INTEGRITY

Consulte el apartado “SQL compuesto (dinámico)” en la página 642 y el apartado “CREATE TRIGGER” en la página 910 para conocer las sentencias de SQL que se permiten en estos contextos.

**Nota:** Puede incluir sentencias CALL en el cuerpo de un procedimiento SQL, pero estas sentencias CALL sólo pueden llamar a otro procedimiento SQL o a un procedimiento C. Las sentencias CALL incluidas en el cuerpo de un procedimiento SQL no pueden llamar a otros tipos de procedimientos almacenados.

## Sentencia ALLOCATE CURSOR

---

### Sentencia ALLOCATE CURSOR

La sentencia ALLOCATE CURSOR asigna un cursor para el conjunto resultante identificado por la variable localizadora de conjuntos resultantes. Vea “Sentencia ASSOCIATE LOCATORS” en la página 1216 para obtener más información sobre las variables localizadoras de conjuntos resultantes.

#### Sintaxis

►—ALLOCATE—*nombre-cursor*—CURSOR FOR RESULT SET—*variable-localizadora-cr*—►

#### Descripción

*nombre-cursor*

Indica el cursor. El nombre no debe identificar un cursor que ya esté declarado en el procedimiento SQL fuente (SQLSTATE 24502).

**CURSOR FOR RESULT SET** *variable-localizadora-cr*

Indica una variable localizadora de conjuntos de resultados que se ha declarado en el procedimiento SQL fuente, de acuerdo con las reglas para las variables del lenguaje principal. Para obtener más información sobre la declaración de variables de SQL, vea “declaración de variables SQL” en la página 1234.

La variable localizadora de conjuntos resultantes debe contener un valor válido, tal como lo devuelve la sentencia ASSOCIATE LOCATORS de SQL (SQLSTATE 24501).

#### Notas

- **Sentencias ALLOCATE CURSOR preparadas dinámicamente:** Se debe utilizar la sentencia EXECUTE junto con la cláusula USING para ejecutar una sentencia ALLOCATE CURSOR preparada dinámicamente. En una sentencia preparada dinámicamente, las referencias a variables del lenguaje principal están representadas por marcadores de parámetros (signos de interrogación).  
En la sentencia ALLOCATE CURSOR, *variable-localizadora-cr* es siempre una variable del lenguaje principal. Por lo tanto, para una sentencia ALLOCATE CURSOR preparada dinámicamente, la cláusula USING de la sentencia EXECUTE debe identificar la variable del lenguaje principal cuyo valor sustituirá al marcador de parámetros representativo de *variable-localizadora-cr*.
- No se puede preparar una sentencia ALLOCATE CURSOR con un identificador de sentencia que ya se ha utilizado en una sentencia DECLARE CURSOR. Por ejemplo, las sentencias de SQL siguientes no son válidas porque la sentencia PREPARE utiliza STMT1 como identificador de la sentencia ALLOCATE CURSOR y STMT1 ya se ha utilizado para una sentencia DECLARE CURSOR.

```
DECLARE CURSOR C1 FOR STMT1;

PREPARE STMT1 FROM
 'ALLOCATE C2 CURSOR FOR RESULT SET ?';
```

### Normas

- Son aplicables las reglas siguientes cuando se utiliza un cursor asignado:
  - Un cursor asignado no se puede abrir con la sentencia **OPEN** (SQLSTATE 24502).
  - Un cursor asignado se puede cerrar con la sentencia **CLOSE**. El cierre de un cursor asignado cierra el cursor asociado existente en el procedimiento almacenado.
  - Sólo se puede asignar un único cursor a cada conjunto resultante.
- Los cursores asignados permanecen vigentes hasta que se ejecuta una operación de retrotracción, un cierre implícito o un cierre explícito.
- Las operaciones **COMMIT** destruyen los cursores asignados que no están definidos con **WITH HOLD** por el procedimiento almacenado.
- La destrucción de un cursor asignado cierra el cursor asociado existente en el procedimiento **SQL**.

### Ejemplos

Este ejemplo de procedimiento **SQL** define y asocia el cursor **C1** a una variable localizadora de conjuntos resultantes, **LOC1**, y con el conjunto resultante asociado devuelto por el procedimiento **SQL**:

```
ALLOCATE C1 CURSOR FOR RESULT SET LOC1;
```

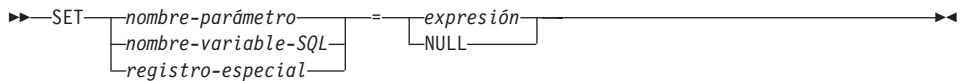
## Sentencia de asignación

---

### Sentencia de asignación

La sentencia de asignación asigna un valor a un parámetro de salida, a una variable local o a un registro especial.

#### Sintaxis



#### Descripción

##### *nombre-parámetro*

Identifica el parámetro que es el sujeto de la asignación. El parámetro se debe especificar en *declaración-parámetro* de la sentencia CREATE PROCEDURE y debe estar definido como parámetro de salida (OUT) o de entrada/salida (INOUT).

##### *nombre-variable-SQL*

Identifica la variable SQL que es el sujeto de la asignación. Las variables SQL se deben declarar antes de utilizarlas. Las variables SQL se pueden definir en una sentencia compuesta.

##### *registro-especial*

Identifica el registro especial que es el sujeto de la asignación. Si el registro especial acepta un nombre de esquema como valor, incluidos los registros especiales CURRENT FUNCTION PATH o CURRENT SCHEMA, DB2 determina si el parámetro de asignación es una variable SQL. Si el parámetro de asignación es una variable SQL, DB2 asigna el valor de la variable SQL al registro especial. Si el parámetro de asignación no es una variable SQL, DB2 considera que el parámetro de asignación es un nombre de esquema y asigna ese nombre al registro especial.

Los valores iniciales de los registros especiales de un procedimiento SQL se heredan del llamador del procedimiento. La asignación de un nuevo valor es válida para el procedimiento SQL completo donde está definido, y será heredado por cualquier procedimiento al que invoque subsiguientemente. Cuando un procedimiento devuelve el control al llamador, los registros especiales se restauran a los valores originales del llamador.

##### *expresión* o NULL

Especifica la expresión o valor que es la fuente de la asignación.

#### Normas

- Las sentencias de asignación de los procedimientos SQL deben ajustarse a las reglas de asignación del SQL.
- El tipo de datos del destino y del fuente deben ser compatibles.

- Cuando se asigna una serie de caracteres a una variable de longitud fija y la longitud de la serie es menor que el atributo de longitud del destino, la serie se rellena por la derecha con el número necesario de blancos (de un solo byte, de doble byte o del juego UCS-2).
- Cuando una serie de caracteres se asigna a una variable y la serie es mayor que el atributo de longitud de la variable, se emite un error.
- Una serie asignada a una variable se convierte primero, si es necesario, a la página de códigos del destino.
- Si se produce un truncamiento de la parte entera del número al asignar a una variable numérica, se emite un error.

### Ejemplos

Incremente la variable SQL `p_salary` en un 10 por ciento.

```
SET p_salary = p_salary + (p_salary * .10)
```

Establezca la variable SQL `p_salary` en el valor nulo.

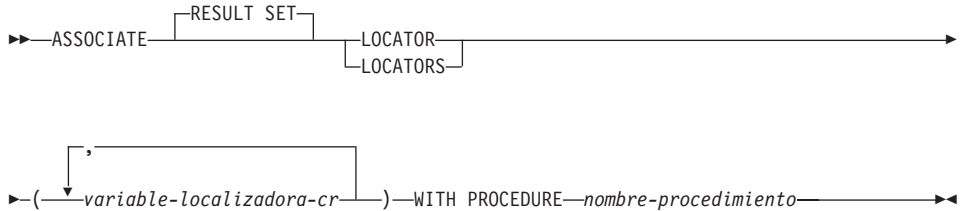
```
SET p_salary = NULL
```

## Sentencia ASSOCIATE LOCATORS

### Sentencia ASSOCIATE LOCATORS

La sentencia ASSOCIATE LOCATORS obtiene el valor localizador de cada conjunto resultante devuelto por un procedimiento almacenado.

#### Sintaxis



#### Descripción

##### *variable-localizadora-cr*

Especifica una variable localizadora de conjuntos resultantes que se ha declarado en una sentencia compuesta.

##### WITH PROCEDURE

Identifica el procedimiento almacenado que devuelve localizadores de conjuntos resultantes de acuerdo con el nombre de procedimiento especificado.

##### *nombre-procedimiento*

Un nombre de procedimiento es un nombre calificado o no calificado. Cada parte del nombre debe estar formada por caracteres SBCS.

Un nombre de procedimiento totalmente calificado consta de dos partes. La primera parte es un identificador que contiene el nombre de esquema del procedimiento almacenado. La última parte es un identificador que contiene el nombre del procedimiento almacenado. Las dos partes deben estar separadas por un punto. Cualquiera de las partes o ambas puede ser un identificador delimitado.

Si el nombre de procedimiento no está calificado, consta de un solo nombre, pues el nombre de esquema implícito no se añade como calificador al nombre del procedimiento. Para que la sentencia ASSOCIATE LOCATOR se ejecute satisfactoriamente sólo es necesario que el nombre de procedimiento no calificado contenido en la sentencia sea el mismo que el nombre de procedimiento contenido en la sentencia CALL ejecutada más recientemente y que se especificó con un nombre de procedimiento no calificado. Cuando se comparan los nombres, no se tiene en cuenta el nombre de esquema implícito del nombre no calificado contenido en la sentencia CALL. A continuación se describen las reglas para especificar un nombre de procedimiento.

Cuando se ejecuta la sentencia ASSOCIATE LOCATORS, el nombre o especificación del procedimiento debe identificar un procedimiento almacenado que el peticionario ya ha invocado utilizando la sentencia CALL. El nombre de procedimiento ASSOCIATE LOCATORS se debe especificar de la misma manera que se especificó en la sentencia CALL. Por ejemplo, si en la sentencia CALL se especificó un nombre de dos partes, se debe utilizar un nombre de dos partes en la sentencia ASSOCIATE LOCATORS.

### Normas

- Se puede asignar más de un localizador a un conjunto resultante. Se puede emitir una misma sentencia ASSOCIATE LOCATORS más de una vez con diferentes variables localizadoras de conjuntos resultantes.
- Si el número de variables localizadoras de conjuntos resultantes que aparecen en la sentencia ASSOCIATE LOCATORS es menor que el número de localizadores devueltos por el procedimiento almacenado, todas las variables de la sentencia se asignan a un valor, y se emite un aviso.
- Si el número de variables localizadoras de conjuntos resultantes que aparecen en la sentencia ASSOCIATE LOCATORS es mayor que el número de localizadores devueltos por el procedimiento almacenado, se asigna el valor 0 a las variables sobrantes.
- Si un mismo llamador invoca un procedimiento almacenado más de una vez, sólo son accesibles los conjuntos resultantes más recientes.

### Ejemplos

En los ejemplos siguientes se da por supuesto que las sentencias utilizadas están intercaladas en procedimientos SQL.

*Ejemplo 1:* Utilice las variables localizadoras de conjuntos resultantes LOC1 y LOC2 para obtener los valores de los dos conjuntos resultantes devueltos por el procedimiento almacenado P1. Se supone que el procedimiento almacenado se invoca utilizando un nombre que consta de dos partes.

```
CALL P1;
ASSOCIATE RESULT SET LOCATORS (LOC1, LOC2)
WITH PROCEDURE P1;
```

*Ejemplo 2:* Repita el supuesto del Ejemplo 1, pero utilice un nombre de dos partes para especificar un nombre de esquema explícito y asegurar que se utilice el procedimiento almacenado P1 del esquema MYSCHEMA.

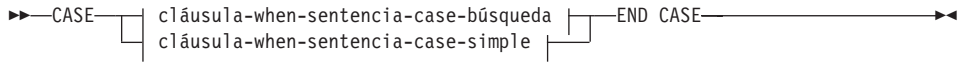
```
CALL MYSCHEMA.P1;
ASSOCIATE RESULT SET LOCATORS (LOC1, LOC2)
WITH PROCEDURE MYSCHEMA.P1;
```

## Sentencia CASE

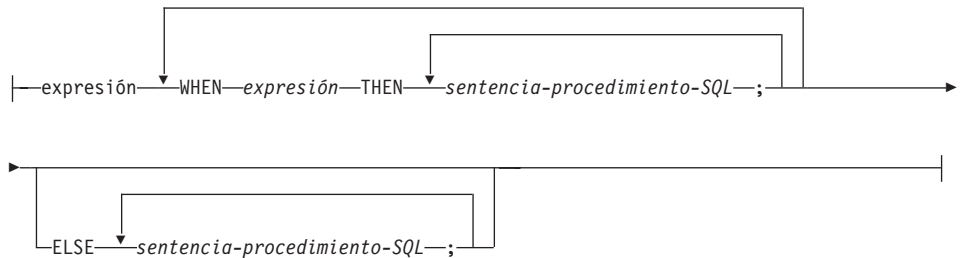
### Sentencia CASE

La sentencia CASE selecciona una vía de ejecución de acuerdo con varias condiciones.

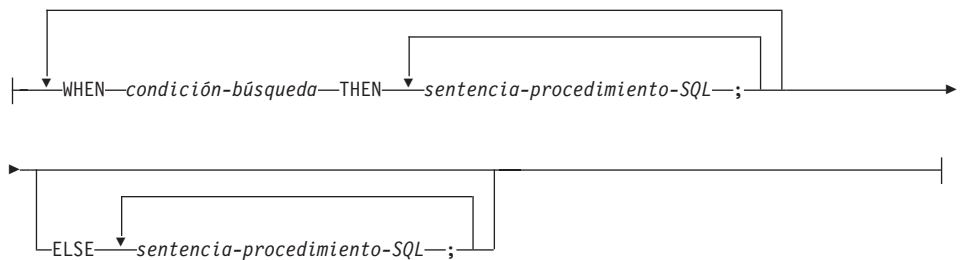
#### Sintaxis



#### cláusula-when-sentencia-case-simple:



#### cláusula-when-sentencia-case-búsqueda:



#### Descripción

##### CASE

Inicia una *expresión-case*.

##### cláusula-when-sentencia-case-simple

El valor de la *expresión* anterior a la primera palabra clave WHEN se comprueba si es igual al valor de cada *expresión* que sigue a la palabra clave WHEN. Si se cumple la condición de búsqueda, se ejecuta la sentencia THEN. Si el resultado es desconocido o falso, el proceso continúa en la siguiente condición de búsqueda. Si el resultado no



coincide con ninguna de las condiciones de búsqueda y existe una cláusula ELSE, se procesan las sentencias de la cláusula ELSE.

### *cláusula-when-sentencia-case-búsqueda*

Se evalúa la *condición-búsqueda* que sigue a la palabra clave WHEN. Si su evaluación da un resultado verdadero, se procesan las sentencias de la cláusula THEN asociada. Si su evaluación da un resultado falso o desconocido, se evalúa la siguiente *condición-búsqueda*. Si ninguna *condición-búsqueda* devuelve un resultado verdadero y existe una cláusula ELSE, se procesan las sentencias de la cláusula ELSE.

### *sentencia-procedimiento-SQL*

Especifica una sentencia que se debe invocar.

### END CASE

Finaliza una *sentencia-case*.

## Notas

- Si ninguna de las condiciones especificadas en la cláusula WHEN devuelve un resultado verdadero y no hay una cláusula ELSE especificada, se emite un error durante la ejecución y se interrumpe la ejecución de la sentencia CASE (SQLSTATE 20000).
- Asegúrese de que la sentencia CASE abarca todas las condiciones de ejecución posibles.

## Ejemplos

De acuerdo con el valor de la variable SQL `v_workdept`, actualice la columna DEPTNAME de la tabla DEPARTMENT utilizando el nombre apropiado.

El ejemplo siguiente muestra cómo hacer esto utilizando la sintaxis de una *cláusula-when-sentencia-case-simple*:

```

CASE v_workdept
 WHEN 'A00'
 THEN UPDATE department
 SET deptname = 'DATA ACCESS 1';
 WHEN 'B01'
 THEN UPDATE department
 SET deptname = 'DATA ACCESS 2';
 ELSE UPDATE department
 SET deptname = 'DATA ACCESS 3';
END CASE

```

El ejemplo siguiente muestra cómo hacer esto utilizando la sintaxis de una *cláusula-when-sentencia-case-búsqueda*:

```

CASE
 WHEN v_workdept = 'A00'
 THEN UPDATE department
 SET deptname = 'DATA ACCESS 1';
 WHEN v_workdept = 'B01'
 THEN UPDATE department

```

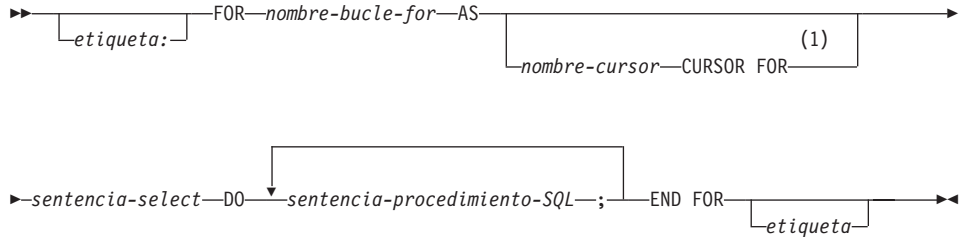
## Sentencia CASE

```
 SET deptname = 'DATA ACCESS 2';
ELSE UPDATE department
 SET deptname = 'DATA ACCESS 3';
END CASE
```

## Sentencia FOR

La sentencia FOR ejecuta una sentencia o grupo de sentencias para cada fila de una tabla.

### Sintaxis



### Notas:

- 1 Esta opción sólo se puede utilizar en el contexto de un procedimiento SQL.

### Descripción

#### *etiqueta*

Especifica la etiqueta de la sentencia FOR. Si se especifica la etiqueta inicial, esa etiqueta puede utilizarse en sentencias LEAVE e ITERATE. Si se especifica la etiqueta final, ésta deberá ser igual que la etiqueta inicial.

#### *nombre-bucle-for*

Especifica una etiqueta para la sentencia compuesta implícita que se genera para implementar la sentencia FOR. Sigue las reglas para la etiqueta de una sentencia compuesta excepto en que no se puede utilizar con una sentencia ITERATE o LEAVE en la sentencia FOR. El *nombre-bucle-for* se utiliza para calificar los nombres de columna devueltos por la *sentencia-select* especificada.

#### *nombre-cursor*

Designa el cursor utilizado para seleccionar filas de la tabla resultante de la sentencia SELECT. Si no se especifica, DB2 genera un nombre de cursor exclusivo.

#### *sentencia-select*

Especifica la sentencia SELECT del cursor. Todas las columnas de la lista de selección deben tener un nombre y no pueden existir dos columnas con el mismo nombre.

En un desencadenante, una función, un método o una sentencia dinámica compuesta, la *sentencia-select* sólo debe constar de una *selección completa* con expresiones de tabla común opcionales.

## Sentencia FOR

### *sentencia-procedimiento-SQL*

Especifica una sentencia (o sentencias) que se debe invocar para cada fila de la tabla.

### Normas

- La lista de selección debe constar de nombres de columna exclusivos y la tabla especificada en la lista de selección debe existir cuando se crea el procedimiento, o debe ser una tabla creada en una sentencia de procedimiento SQL anterior.
- El cursor especificado en una sentencia-for no puede estar referenciado fuera de la sentencia-for y no se puede especificar en una sentencia OPEN, FETCH ni CLOSE.

### Ejemplos

El ejemplo siguiente utiliza la sentencia-for para ejecutar un proceso iterativo sobre la tabla `employee` completa. Para cada fila de la tabla, la variable SQL `fullname` se establece en el primer apellido del empleado, seguido de una coma, el nombre, un espacio en blanco y la inicial del segundo apellido. Cada valor de `fullname` se inserta en la tabla `tnames`.

```
BEGIN
 DECLARE fullname CHAR(40);
 FOR v1 AS
 SELECT firstnme, midinit, lastname FROM employee
 DO
 SET fullname = lastname || ',' || firstnme || ' ' || midinit;
 INSERT INTO tnames VALUE (fullname);
END FOR
END
```

## Sentencia GET DIAGNOSTICS

La sentencia GET DIAGNOSTICS obtiene información sobre la sentencia de SQL anterior invocada.

### Sintaxis

```
► GET DIAGNOSTICS nombre-variable-SQL = { ROW_COUNT | RETURN_STATUS } ►
```

### Descripción

*nombre-variable-SQL*

Identifica la variable que es el sujeto de la asignación. La variable debe ser una variable entera. Las variables SQL se pueden definir en una sentencia compuesta.

#### ROW\_COUNT

Identifica el número de columnas asociadas con la sentencia de SQL anterior invocada. Si la sentencia de SQL anterior es una sentencia DELETE, INSERT o UPDATE, ROW\_COUNT identifica el número de filas suprimidas, insertadas o actualizadas por esa sentencia, excluidas las filas afectadas por desencadenantes o restricciones de integridad referencial. Si la sentencia anterior es una sentencia PREPARE, ROW\_COUNT identifica el número *estimado* de columnas resultantes de la sentencia preparada.

#### RETURN\_STATUS

Identifica el valor de estado devuelto por el procedimiento almacenado asociado a la sentencia de SQL ejecutada anteriormente, siempre que la sentencia fuera una sentencia CALL que invoca un procedimiento que devuelve un estado. Si la sentencia anterior no es de esa clase, el valor devuelto no tiene ningún significado y podría ser un entero cualquiera.

### Normas

- La sentencia GET DIAGNOSTICS no cambia el contenido del área de diagnósticos (SQLCA). Las variables especiales SQLSTATE o SQLCODE declaradas en un procedimiento SQL se establecen en el SQLSTATE o SQLCODE devuelto por la ejecución de la sentencia GET DIAGNOSTICS.

### Ejemplos

En un procedimiento SQL, ejecute una sentencia GET DIAGNOSTICS para determinar cuántas filas se actualizaron.

```
CREATE PROCEDURE sqlproc (IN deptnbr VARCHAR(3))
LANGUAGE SQL
BEGIN
 DECLARE SQLSTATE CHAR(5);
 DECLARE rcount INTEGER;
 UPDATE CORPDATA.PROJECT
SET PRSTAFF = PRSTAFF + 1,5
```

## Sentencia GET DIAGNOSTICS

```
 WHERE DEPTNO = deptnbr;
 GET DIAGNOSTICS rcount = ROW_COUNT;
-- En este momento, rcount contiene el número de filas que se actualizaron.
 ...
 END
```

Dentro de un procedimiento SQL, procese el valor de estado devuelto por la invocación de un procedimiento almacenado llamado TRYIT, el cual puede devolver explícitamente un valor positivo, lo que indica un error de usuario, o encontrar errores SQL que producirían un valor de estado de retorno negativo. Si el procedimiento se ejecuta satisfactoriamente, devuelve un valor igual a cero.

```
CREATE PROCEDURE TESTIT ()
LANGUAGE SQL
A1:BEGIN
 DECLARE RETVAL INTEGER DEFAULT 0;
 ...
 CALL TRYIT;
 GET DIAGNOSTICS RETVAL = RETURN_STATUS;
 IF RETVAL <> 0 THEN
 ...
 LEAVE A1;
 ELSE
 ...
 END IF;
 END A1
```

## Sentencia GOTO

La sentencia GOTO se utiliza para bifurcar hacia una etiqueta definida por el usuario dentro de una rutina SQL.

### Sintaxis

►►—GOTO—*etiqueta*—◄◄

### Descripción

*etiqueta*

Especifica una sentencia con etiqueta donde debe continuar el proceso. La sentencia con etiqueta y la sentencia GOTO deben estar en el mismo ámbito:

- Si la sentencia GOTO se define en una sentencia FOR, la *etiqueta* se debe definir dentro de la misma sentencia FOR, que no sea una sentencia FOR anidada ni una sentencia compuesta anidada.
- Si la sentencia GOTO se define en una sentencia compuesta, la *etiqueta* se debe definir dentro de la misma sentencia compuesta, que no sea una sentencia FOR anidada ni una sentencia compuesta anidada.
- Si la sentencia GOTO se define en un gestor de condiciones, la *etiqueta* se debe definir en el mismo gestor de condiciones, de acuerdo con las demás reglas sobre el ámbito
- Si la sentencia GOTO se define fuera de un gestor de condiciones, la *etiqueta* no se debe definir dentro de un gestor de condiciones.

Si la *etiqueta* no está definida dentro de un ámbito que sea accesible para la sentencia GOTO, se produce un error (SQLSTATE 42736).

### Normas

- Es aconsejable utilizar la sentencia GOTO de forma moderada. Esta sentencia interfiere en las secuencias normales de proceso, lo que hace que la rutina sea más difícil de leer y actualizar. Antes de utilizar una sentencia GOTO, determine si en su lugar puede utilizarse otra sentencia, tal como IF o LEAVE, para eludir la necesidad de utilizar una sentencia GOTO.

### Ejemplos

En la sentencia compuesta siguiente, los parámetros *rating* y *v\_empno* se pasan al procedimiento, el cual entonces devuelve el parámetro de salida *return\_parm* en forma de duración de fecha. Si el tiempo de servicio del empleado en la empresa es menor que 6 meses, la sentencia GOTO transfiere el control al final del procedimiento, y el valor *new\_salary* permanece inalterado.

## Sentencia GOTO

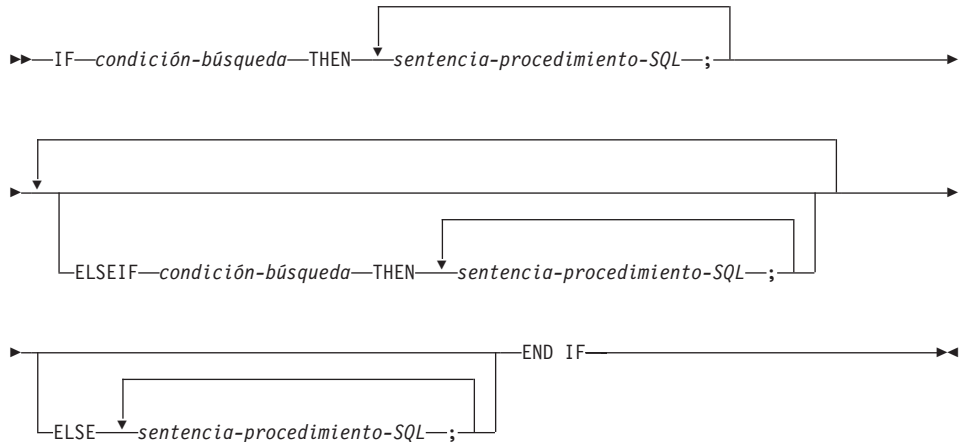
```
CREATE PROCEDURE adjust_salary
(IN v_empno CHAR(6),
 IN rating INTEGER)
OUT return_parm DECIMAL (8,2)
MODIFIES SQL DATA
LANGUAGE SQL
BEGIN
 DECLARE new_salary DECIMAL (9,2)
 DECLARE service DECIMAL (8,2)
 SELECT SALARY, CURRENT_DATE - HIREDATE
 INTO new_salary, service
 FROM EMPLOYEE
 WHERE EMPNO = v_empno
 IF service < 600
 THEN GOTO EXIT
 END IF
 IF rating = 1
 THEN SET new_salary = new_salary + (new_salary * .10)
 ELSE IF rating = 2
 THEN SET new_salary = new_salary + (new_salary * .05)
 END IF
 UPDATE EMPLOYEE
 SET SALARY = new_salary
 WHERE EMPNO = v_empno
 EXIT: SET return_parm = service
END
```



## Sentencia IF

La sentencia IF selecciona una vía de ejecución de acuerdo con la evaluación de una condición.

### Sintaxis



### Descripción

#### *condición-búsqueda*

Especifica la condición para la cual debe invocarse una sentencia de SQL. Si la condición es desconocida o falsa, el proceso continúa en la siguiente condición de búsqueda, hasta que una condición sea verdadera o el proceso llegue a la cláusula ELSE.

#### *sentencia-procedimiento-SQL*

Especifica la sentencia que se debe invocar si la condición de búsqueda anterior es verdadera. Si la evaluación de ninguna condición de búsqueda da un resultado verdadero, se invoca la *sentencia-procedimiento-SQL* que sigue a la palabra clave ELSE.

### Ejemplos

El procedimiento SQL siguiente acepta dos parámetros de entrada: un número de empleado (*employee\_number*) y una tarifa de salario (*rating*). Dependiendo del valor de *rating*, la tabla *employee* se actualiza con nuevos valores en las columnas *salary* (salario) y *bonus* (prima).

```

CREATE PROCEDURE UPDATE_SALARY_IF
 (IN employee_number CHAR(6), INOUT rating SMALLINT)
 LANGUAGE SQL
 BEGIN
 DECLARE not_found CONDITION FOR SQLSTATE '02000';
 DECLARE EXIT HANDLER FOR not_found
 SET rating = -1;

```

## Sentencia IF

```
IF rating = 1
 THEN UPDATE employee
 SET salary = salary * 1.10, bonus = 1000
 WHERE empno = employee_number;
ELSEIF rating = 2
 THEN UPDATE employee
 SET salary = salary * 1.05, bonus = 500
 WHERE empno = employee_number;
ELSE UPDATE employee
 SET salary = salary * 1.03, bonus = 0
 WHERE empno = employee_number;
END IF;
END
```

## Sentencia ITERATE

La sentencia ITERATE hace que el flujo de control vuelva al principio de un bucle con etiqueta.

### Sintaxis

►►—ITERATE—*etiqueta*—◄◄

### Descripción

*etiqueta*

Especifica la etiqueta de la sentencia FOR, LOOP, REPEAT o WHILE a la cual DB2 transfiere el flujo de control.

### Ejemplos

Este ejemplo utiliza un cursor para devolver información para un nuevo departamento. Si se ha invocado el gestor de condiciones *not\_found*, el flujo de control elude el bucle. Si el valor de *v\_dept* es 'D11', la sentencia ITERATE devuelve el flujo de control al principio de la sentencia LOOP. En otro caso, se inserta una nueva fila en la tabla DEPARTMENT.

```
CREATE PROCEDURE ITERATOR()
LANGUAGE SQL
BEGIN
 DECLARE v_dept CHAR(3);
 DECLARE v_deptname VARCHAR(29);
 DECLARE v_admdept CHAR(3);
 DECLARE at_end INTEGER DEFAULT 0;
 DECLARE not_found CONDITION FOR SQLSTATE '02000';
 DECLARE c1 CURSOR FOR
 SELECT deptno, deptname, admrdept
 FROM department
 ORDER BY deptno;
 DECLARE CONTINUE HANDLER FOR not_found
 SET at_end = 1;
 OPEN c1;
 ins_loop:
 LOOP
 FETCH c1 INTO v_dept, v_deptname, v_admdept;
 IF at_end = 1 THEN
 LEAVE ins_loop;
 ELSEIF v_dept = 'D11' THEN
 ITERATE ins_loop;
 END IF;
 INSERT INTO department (deptno, deptname, admrdept)
 VALUES ('NEW', v_deptname, v_admdept);
 END LOOP;
 CLOSE c1;
END
```

## Sentencia LEAVE

---

### Sentencia LEAVE

La sentencia LEAVE transfiere el control del programa hacia fuera de un bucle o de una sentencia compuesta.

#### Sintaxis

►►—LEAVE—*etiqueta*—◄◄

#### Descripción

*etiqueta*

Especifica la etiqueta de la sentencia FOR, LOOP, REPEAT, WHILE o sentencia compuesta cuya ejecución debe concluir.

#### Normas

Cuando una sentencia LEAVE transfiere el control hacia fuera de una sentencia compuesta, se cierran todos los cursores abiertos de la sentencia compuesta, excepto los cursores utilizados para devolver conjuntos resultantes.

#### Ejemplos

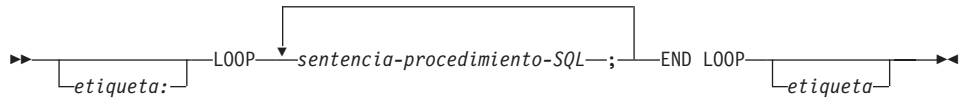
Este ejemplo contiene un bucle que lee datos para el cursor *c1*. Si el valor de la variable SQL *at\_end* no es cero, la sentencia LEAVE transfiere el control hacia fuera del bucle.

```
CREATE PROCEDURE LEAVE_LOOP(OUT counter INTEGER)
LANGUAGE SQL
BEGIN
 DECLARE v_counter INTEGER;
 DECLARE v_firstnme VARCHAR(12);
 DECLARE v_midinit CHAR(1);
 DECLARE v_lastname VARCHAR(15);
 DECLARE at_end SMALLINT DEFAULT 0;
 DECLARE not_found CONDITION FOR SQLSTATE '02000';
 DECLARE c1 CURSOR FOR
 SELECT firstnme, midinit, lastname
 FROM employee;
 DECLARE CONTINUE HANDLER for not_found
 SET at_end = 1;
 SET v_counter = 0;
 OPEN c1;
fetch_loop:
 LOOP
 FETCH c1 INTO v_firstnme, v_midinit, v_lastname;
 IF at_end <> 0 THEN LEAVE fetch_loop;
 END IF;
 SET v_counter = v_counter + 1;
 END LOOP fetch_loop;
 SET counter = v_counter;
 CLOSE c1;
END
```

## Sentencia LOOP

La sentencia LOOP repite la ejecución de una sentencia o grupo de sentencias.

### Sintaxis



### Descripción

*etiqueta*

Especifica la etiqueta de la sentencia LOOP. Si se especifica la etiqueta inicial, esa etiqueta puede especificarse en sentencias LEAVE e ITERATE. Si se especifica la etiqueta final, se debe especificar la etiqueta inicial correspondiente.

*sentencia-procedimiento-SQL*

Especifica las sentencias que se deben invocar en el bucle.

### Ejemplos

Este procedimiento utiliza una sentencia LOOP para leer valores de la tabla employee. Cada vez que se repite el bucle, se incrementa el parámetro de salida *counter* y se comprueba el valor de *v\_midinit* para asegurarse de que ese valor no es un espacio en blanco individual (' '). Si *v\_midinit* es un espacio en blanco individual, la sentencia LEAVE transfiere el flujo de control hacia fuera del bucle.

```

CREATE PROCEDURE LOOP_UNTIL_SPACE(OUT counter INTEGER)
LANGUAGE SQL
BEGIN
 DECLARE v_counter INTEGER DEFAULT 0;
 DECLARE v_firstnme VARCHAR(12);
 DECLARE v_midinit CHAR(1);
 DECLARE v_lastname VARCHAR(15);
 DECLARE c1 CURSOR FOR
 SELECT firstnme, midinit, lastname
 FROM employee;
 DECLARE CONTINUE HANDLER FOR NOT FOUND
 SET counter = -1;
 OPEN c1;
fetch_loop:
 LOOP
 FETCH c1 INTO v_firstnme, v_midinit, v_lastname;
 IF v_midinit = ' ' THEN
 LEAVE fetch_loop;
 END IF;
 SET v_counter = v_counter + 1;

```

## Sentencia LOOP

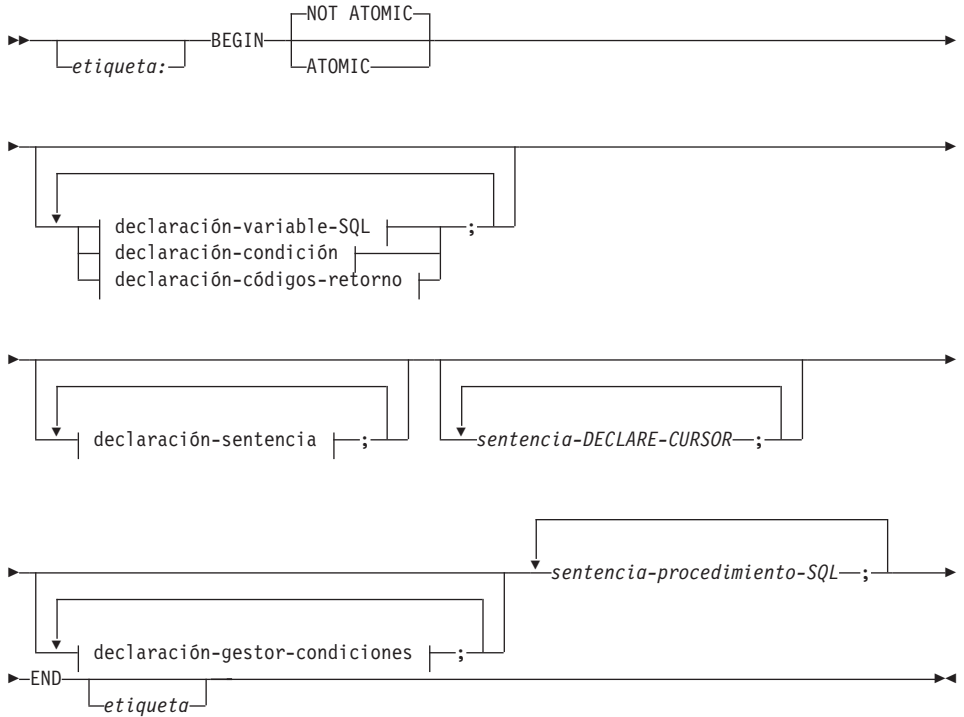
```
END LOOP fetch_loop;
SET counter = v_counter;
CLOSE c1;
END
```

**Sentencia compuesta (procedimiento)**

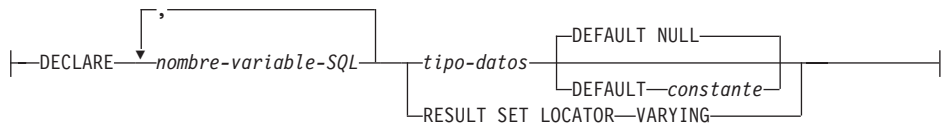
Una sentencia compuesta de procedimiento agrupa otras sentencias en un procedimiento SQL. Dentro de una sentencia compuesta se pueden declarar variables SQL, cursores y gestores de condiciones.

**Sintaxis**

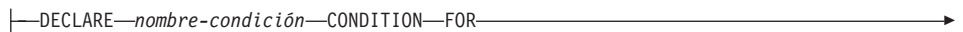
**sentencia-compuesta-procedimiento**



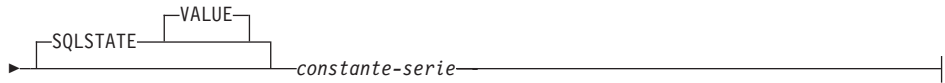
**declaración-variable-SQL:**



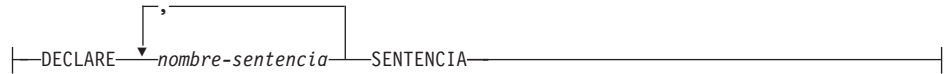
**declaración-condición:**



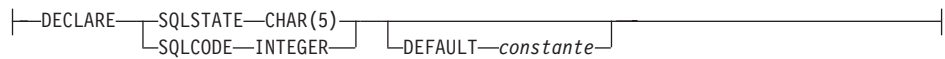
## Sentencia compuesta (procedimiento)



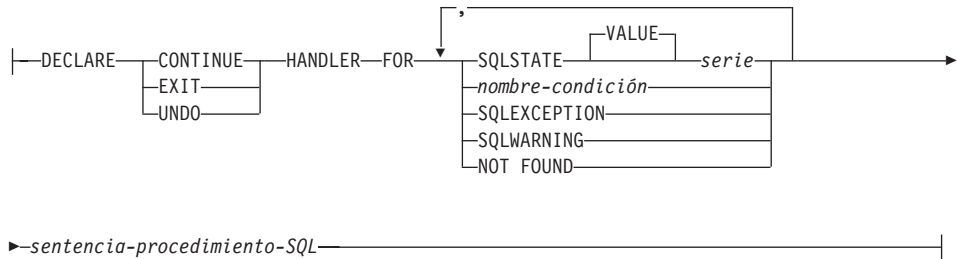
### declaración-sentencia:



### declaración-códigos-retorno:



### declaración-gestor-condiciones:



## Descripción

### etiqueta

Define la etiqueta del bloque de programa. Si se especifica la etiqueta inicial, se puede utilizar para calificar variables SQL declaradas en la sentencia compuesta y también se puede especificar en una sentencia LEAVE. Si se especifica la etiqueta final, ésta deberá ser igual que la etiqueta inicial.

### ATOMIC o NOT ATOMIC

ATOMIC indica que si se produce un error en la sentencia compuesta, se retrotraen todas las sentencias de SQL de la sentencia compuesta. NOT ATOMIC indica que un error dentro de la sentencia compuesta no produce la retrotracción de la sentencia.

### declaración-variable-SQL

Declara una variable que es local respecto a la sentencia compuesta.



### *nombre-variable-SQL*

Define el nombre de una variable local. DB2 convierte a mayúsculas todos los nombres de variables del SQL. El nombre no puede ser el mismo que otra variable SQL existente en la misma sentencia compuesta y no puede ser igual que un nombre de parámetro. Los nombres de variables SQL no deben ser iguales que los nombres de columnas. Si una sentencia de SQL contiene un identificador que tiene el mismo nombre que una variable SQL y una referencia a columna, DB2 interpreta el identificador como una columna.

### *tipo-datos*

Especifica el tipo de datos de la variable. Vea “Tipos de datos” en la página 82 para obtener una descripción de los tipos de datos de SQL. No se da soporte a los tipos de datos definidos por el usuario, los tipos gráficos ni a FOR BIT DATA.

### **DEFAULT** *constante* o **NULL**

Define el valor por omisión de la variable SQL. La variable se inicializa cuando se invoca el procedimiento SQL. Si no se especifica un valor por omisión, el valor de la variable se inicializa en NULL.

### **RESULT\_SET\_LOCATOR VARYING**

Especifica el tipo de datos de una variable localizadora de conjuntos resultantes.

### **declaración-condición**

Declara el nombre de una condición y el valor SQLSTATE asociado.

### *nombre-condición*

Especifica el nombre de la condición. El nombre de la condición debe ser exclusivo dentro del cuerpo del procedimiento y sólo puede estar referenciado dentro de la sentencia compuesta donde está declarado.

### **FOR SQLSTATE** *constante-serie*

Especifica el SQLSTATE que está asociado a la condición. La constante-serie se debe especificar en forma de cinco caracteres encerrados entre comillas simples y no puede ser '00000'.

### **declaración-sentencia**

Declara una lista de uno o más nombres que son locales en la sentencia compuesta. Un nombre de sentencia no puede ser igual que otro nombre de sentencia en la misma sentencia compuesta.

### **declaración-códigos-retorno**

Declara las variables especiales llamadas SQLSTATE y SQLCODE, que se establecen automáticamente en el valor que se devuelve tras procesar una sentencia de SQL. Las variables SQLSTATE y SQLCODE sólo se pueden

## Sentencia compuesta (procedimiento)

declarar en la sentencia compuesta más externa del cuerpo del procedimiento SQL. Estas variables sólo se pueden declarar una vez para cada procedimiento SQL.

### *sentencia-declare-cursor*

Declara un cursor en el cuerpo del procedimiento. Cada cursor debe tener un nombre exclusivo. El cursor sólo puede estar referenciado dentro de la sentencia compuesta. Utilice una sentencia OPEN para abrir el cursor, y una sentencia FETCH para leer filas utilizando el cursor. Para que el procedimiento SQL devuelva conjuntos resultantes a la aplicación cliente, el cursor se debe declarar utilizando la cláusula WITH RETURN. El ejemplo siguiente devuelve un conjunto resultante a la aplicación cliente:

```
CREATE PROCEDURE RESULT_SET()
LANGUAGE SQL
RESULT SETS 1
BEGIN
 DECLARE C1 CURSOR WITH RETURN FOR
 SELECT id, name, dept, job
 FROM staff;
 OPEN C1;
END
```

**Nota:** Para procesar los conjuntos resultantes, debe escribir la aplicación cliente utilizando una de las siguientes interfaces de programación de aplicaciones: Interfaz a nivel de llamada de DB2 (CLI de DB2), Open Database Connectivity (ODBC), Java Database Connectivity (JDBC) o SQL incorporado para Java (SQLJ).

Para obtener más información sobre la declaración de un cursor, consulte "DECLARE CURSOR" en la página 976.

### **declaración-gestor-condiciones**

Especifica un *gestor de condiciones*, un conjunto de sentencias que se ejecutan cuando se produce una condición de excepción o terminación en la sentencia compuesta. *sentencia-procedimiento-SQL* es una sentencia que se ejecuta cuando el gestor de condiciones recibe el control.

Un gestor de condiciones sólo está activo dentro de la sentencia compuesta en la que está declarado.

Existen tres tipos de gestores de condiciones:

#### **CONTINUE**

Tras la invocación satisfactoria del gestor de condiciones, el control pasa a la sentencia de SQL que sigue a continuación de la sentencia que provocó la condición de excepción. Si el error que causó la excepción es una sentencia FOR, IF, CASE, WHILE o REPEAT (pero no una sentencia de procedimiento SQL incluida dentro de una de aquéllas), el control pasa a la sentencia que sigue a continuación de END FOR, END IF, END CASE, END WHILE o END REPEAT.

### EXIT

Tras la invocación satisfactoria del gestor de condiciones, el control se transfiere al final de la sentencia compuesta donde se declaró el gestor de condiciones.

### UNDO

Antes de invocar el gestor de condiciones, se retrotraen los cambios de SQL que se hicieron en la sentencia compuesta. Tras la invocación satisfactoria del gestor de condiciones, el control se transfiere al final de la sentencia compuesta donde se declaró el gestor de condiciones. Si se especifica UNDO, se debe especificar ATOMIC.

Situaciones en las que se activa el gestor de condiciones:

#### SQLSTATE *serie*

Especifica un SQLSTATE para el cual se invoca el gestor de condiciones. El SQLSTATE no puede ser '00000'.

#### *nombre-condición*

Especifica una condición para la cual se invoca el gestor de condiciones. El nombre de la condición debe estar definido previamente en una declaración de condición.

#### SQLEXCEPTION

Especifica que el gestor de condiciones se invoca cuando se produce una excepción de SQL (SQLEXCEPTION). Una excepción de SQL es un estado de SQL (SQLSTATE) en el que los dos primeros caracteres no son "00", "01" ni "02".

#### SQLWARNING

Especifica que el gestor de condiciones se invoca cuando se produce un aviso de SQL (SQLWARNING). Un aviso de SQL es un estado de SQL (SQLSTATE) en el que los dos primeros caracteres son "01".

#### NOT FOUND

Especifica que el gestor de condiciones se invoca cuando se produce una condición de "no encontrado" (NOT FOUND). NOT FOUND es un estado de SQL (SQLSTATE) en el que los dos primeros caracteres son "02".

## Normas

- Las sentencias compuestas definidas como ATOMIC no se pueden anidar.
- Las normas siguientes son aplicables a la declaración de un gestor de condiciones:
  - Una declaración de gestor de condiciones que contenga SQLEXCEPTION, SQLWARNING o NOT FOUND no puede contener otros estados de SQL ni nombres de condición.

## Sentencia compuesta (procedimiento)

- Las declaraciones de gestores de condiciones incluidas en una misma sentencia compuesta no pueden contener condiciones duplicadas.
- Una declaración de gestor de condiciones no puede contener el mismo código de condición ni valor de SQLSTATE más de una vez, y no puede contener un valor de SQLSTATE y un nombre de condición que representen el mismo valor de SQLSTATE. Para obtener una lista de los valores de SQLSTATE y más información, consulte el manual *Consulta de mensajes*.
- Un gestor de condiciones se activa cuando es el gestor más apropiado para una condición de excepción o terminación. El gestor de condiciones más apropiado (para la condición de excepción o terminación) es un gestor que está definido en la sentencia compuesta, y cuyo el ámbito es el más cercano a la sentencia que contiene la condición de excepción o terminación. Si se produce una excepción para la que no existe ningún gestor de condiciones, se interrumpe la ejecución de la sentencia compuesta.

### Ejemplos

Creación de un cuerpo de procedimiento con una sentencia compuesta que ejecuta las acciones siguientes:

1. Declara variables SQL
2. Declara un cursor para que proporcione el salario de los empleados de un departamento determinado de acuerdo con un parámetro IN. En la sentencia SELECT, convierte de DECIMAL a DOUBLE el tipo de datos de la columna *salary* (salario).
3. Declara un gestor de condiciones EXIT para la condición NOT FOUND (fin de archivo), que asigna el valor '6666' al parámetro de salida medianSalary (salario medio)
4. Selecciona el número de empleados del departamento especificado y lo coloca en la variable SQL numRecords
5. Lee filas desde el cursor en un bucle WHILE hasta llegar al 50% + 1 de los empleados
6. Devuelve el salario medio

```
CREATE PROCEDURE DEPT_MEDIAN
(IN deptNumber SMALLINT, OUT medianSalary DOUBLE)
LANGUAGE SQL
BEGIN
 DECLARE v_numRecords INTEGER DEFAULT 1;
 DECLARE v_counter INTEGER DEFAULT 0;
 DECLARE c1 CURSOR FOR
 SELECT CAST(salary AS DOUBLE) FROM staff
 WHERE DEPT = deptNumber
 ORDER BY salary;
 DECLARE EXIT HANDLER FOR NOT FOUND
 SET medianSalary = 6666;
 -- inicializar parámetro de salida
```

## Sentencia compuesta (procedimiento)

```
SET medianSalary = 0;
SELECT COUNT(*) INTO v_numRecords FROM staff
 WHERE DEPT = deptNumber;
OPEN c1;
WHILE v_counter < (v_numRecords / 2 + 1) DO
 FETCH c1 INTO medianSalary;
 SET v_counter = v_counter + 1;
END WHILE;
CLOSE c1;
END
```

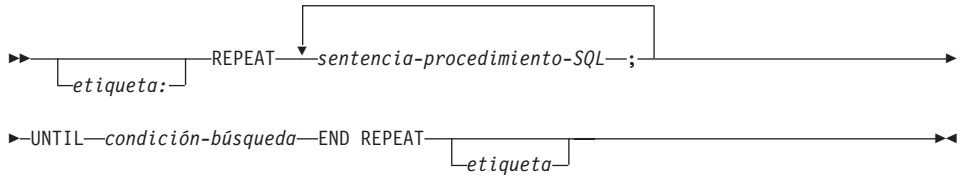
## Sentencia REPEAT

---

### Sentencia REPEAT

La sentencia REPEAT ejecuta una sentencia o grupo de sentencias hasta que se cumpla una condición de búsqueda.

#### Sintaxis



#### Descripción

##### *etiqueta*

Especifica la etiqueta de la sentencia REPEAT. Si se especifica la etiqueta inicial, esa etiqueta puede especificarse en sentencias LEAVE e ITERATE. Si se especifica una etiqueta final, se debe especificar también la etiqueta inicial correspondiente.

##### *sentencia-procedimiento-SQL*

Especifica la sentencia de SQL que se debe ejecutar con el bucle.

##### *condición-búsqueda*

Especifica una condición que se evalúa antes de ejecutarse la sentencia del procedimiento SQL. Si la condición es falsa, DB2 ejecuta la sentencia del procedimiento SQL incluida en el bucle.

#### Ejemplos

En este ejemplo, una sentencia REPEAT lee filas de una tabla hasta que se invoca el gestor de condiciones *not\_found*.

```
CREATE PROCEDURE REPEAT_STMT(OUT counter INTEGER)
LANGUAGE SQL
BEGIN
 DECLARE v_counter INTEGER DEFAULT 0;
 DECLARE v_firstnme VARCHAR(12);
 DECLARE v_midinit CHAR(1);
 DECLARE v_lastname VARCHAR(15);
 DECLARE at_end SMALLINT DEFAULT 0;
 DECLARE not_found CONDITION FOR SQLSTATE '02000';
 DECLARE c1 CURSOR FOR
 SELECT firstnme, midinit, lastname
 FROM employee;
 DECLARE CONTINUE HANDLER FOR not_found
 SET at_end = 1;
 OPEN c1;
 fetch_loop:
 REPEAT
```

## Sentencia REPEAT

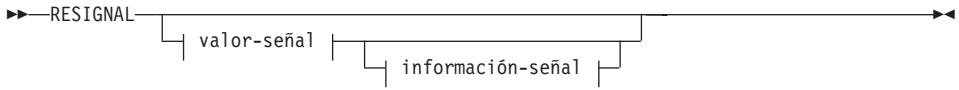
```
 FETCH c1 INTO v_firstnme, v_midinit, v_lastname;
 SET v_counter = v_counter + 1;
 UNTIL at_end > 0
END REPEAT fetch_loop;
SET counter = v_counter;
CLOSE c1;
END
```

## Sentencia RESIGNAL

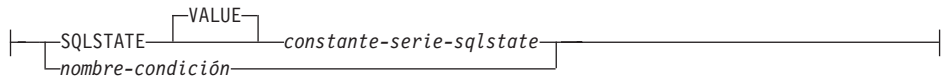
### Sentencia RESIGNAL

La sentencia RESIGNAL se utiliza para retransmitir una condición de error o de aviso. Hace que se devuelva un error o aviso con el SQLSTATE especificado, junto con texto de mensaje opcional.

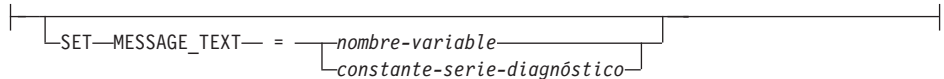
#### Sintaxis



#### valor-señal:



#### información-señal:



#### Descripción

##### SQLSTATE VALUE *constante-serie-sqlstate*

La constante especificada de tipo serie representa un estado SQL (SQLSTATE). Debe ser una constante de tipo serie con exactamente 5 caracteres que siguen las reglas aplicables a los SQLSTATE:

- Cada carácter debe pertenecer al conjunto de dígitos (del '0' al '9') o de letras mayúsculas no acentuadas (de la 'A' a la 'Z')
- La clase SQLSTATE (primeros dos caracteres) no puede ser '00', pues esto representa una finalización satisfactoria.

Si SQLSTATE no se ajusta a estas reglas, se produce un error (SQLSTATE 428B3).

##### *nombre-condición*

Especifica el nombre de la condición.

##### SET MESSAGE\_TEXT =

Especifica una serie de caracteres que describe el error o aviso. La serie de caracteres se coloca en el campo SQLERRMC de la SQLCA. Si la serie tiene más de 70 bytes de longitud, se truncará sin avisar de ello. Esta



cláusula sólo puede especificarse si también está especificado un SQLSTATE o *nombre-condición* (SQLSTATE 42601).

### *nombre-variable*

Identifica una variable SQL que se debe declarar dentro de la sentencia compuesta. La variable SQL debe estar definida con el tipo de datos CHAR o VARCHAR.

### *constante-serie-diagnóstico*

Especifica una constante de tipo serie que contiene el texto del mensaje.

## Notas

- Si la sentencia RESIGNAL se especifica sin una cláusula SQLSTATE o un *nombre-condición*, la rutina SQL devuelve el control al llamador con la misma condición con la que se invocó al gestor de condiciones.
- Si se emite una sentencia RESIGNAL y ésta especifica un SQLSTATE o *nombre-condición*, el código SQL (SQLCODE) asignado es:

+438 si el SQLSTATE comienza con '01' o '02'

-438 en otro caso

Si se emite una sentencia RESIGNAL sin ninguna opción, el SQLCODE permanece inalterado respecto a la excepción que originó la invocación del gestor de condiciones.

- Si el SQLSTATE o condición indica que se señala una excepción (clase de SQLSTATE distinta de '01' y '02'):
  - entonces, se procesa la excepción y el control se transfiere a un gestor de condiciones, siempre que exista un gestor en la siguiente sentencia compuesta externa (o en una sentencia compuesta más externa todavía) respecto a la sentencia compuesta donde reside el gestor de condiciones con la sentencia RESIGNAL, y la sentencia compuesta contenga un gestor de condiciones para el SQLSTATE, nombre-condición o SQLEXCEPTION especificados;
  - en otro caso, la excepción no se procesa y el control se transfiere inmediatamente al final de la sentencia compuesta.
- Si el SQLSTATE o condición indica que se señala un aviso (clase de SQLSTATE '01') o condición "not found" (clase de SQLSTATE '02'):
  - entonces, se procesa el aviso o condición "not found" y el control se transfiere a un gestor de condiciones, siempre que exista un gestor en la siguiente sentencia compuesta externa (o en una sentencia compuesta más externa todavía) respecto a la sentencia compuesta donde reside el gestor de condiciones con la sentencia RESIGNAL, y la sentencia compuesta contenga un gestor de condiciones para el SQLSTATE, nombre-condición, SQLWARNING (si la clase de SQLSTATE es '01') o NOT FOUND (si la clase de SQLSTATE es '02');

## Sentencia RESIGNAL

- en otro caso, el aviso no se procesa y el proceso continúa en la sentencia siguiente.

### Ejemplos

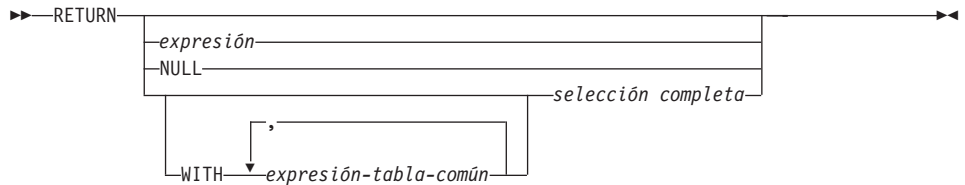
Este ejemplo detecta los errores de división por cero. La sentencia IF utiliza una sentencia SIGNAL para invocar el gestor de condiciones *overflow*. El gestor de condiciones utiliza una sentencia RESIGNAL para devolver un valor de SQLSTATE diferente a la aplicación cliente.

```
CREATE PROCEDURE divide (IN numerator INTEGER
 IN denominator INTEGER
 OUT result INTEGER
LANGUAGE SQL
CONTAINS SQL
BEGIN
 DECLARE overflow CONDITION FOR SQLSTATE '22003';
 DECLARE CONTINUE HANDLER FOR overflow
 RESIGNAL SQLSTATE'22375' ;
 IF denominator = 0 THEN
 SIGNAL overflow;
 ELSE
 SET result = numerator / denominator;
 END IF;
 END
```

## Sentencia RETURN

La sentencia RETURN se utiliza para concluir la ejecución de la rutina. Para las funciones o métodos de SQL, devuelve el resultado de la función o método. Para un procedimiento SQL, devuelve opcionalmente un valor de estado de tipo entero.

### Sintaxis



### Descripción

#### *expresión*

Especifica un valor devuelto procedente de la rutina:

- Si la rutina es una función o un método, se deberá especificar *expresión*, NULL o *selección completa* (SQLSTATE 42630) y el tipo de datos del resultado debe ser asignable al tipo RETURNS de la rutina (SQLSTATE 42866).
- No se puede especificar una expresión escalar (que no sea una selección completa escalar) para una función de tabla (SQLSTATE 428F1).
- Si la rutina es un procedimiento, el tipo de datos de *expresión* debe ser INTEGER (SQLSTATE 428E2). Un procedimiento no puede devolver NULL o una *selección completa*.

#### NULL

Especifica que la función o el método devuelve un valor nulo del tipo de datos definido en la cláusula RETURNS. No se puede especificar NULL para un RETURN de un procedimiento.

#### WITH *expresión-tabla-común*

Define una expresión de tabla común para utilizarla con la *selección completa* que viene a continuación. Vea “expresión-común-tabla” en la página 519.

#### *selección completa*

Especifica la fila o filas que se deben devolver para la función. El número de columnas de la *selección completa* debe coincidir con el número de columnas del resultado de la función (SQLSTATE 42811). Además, los tipos de columnas estáticas de la *selección completa* deben

## Sentencia RETURN

ser asignables a los tipos de columnas declaradas del resultado de la función, utilizando las reglas para la asignación a columnas (SQLSTATE 42866).

No se puede especificar la *selección completa* para un RETURN de un procedimiento.

Si la rutina es una función escalar o un método, la *selección completa* debe devolver una columna (SQLSTATE 42823) y, como máximo, una fila (SQLSTATE 21000).

Si la rutina es una función de fila, debe devolver, como máximo, una fila (SQLSTATE 21505). Sin embargo, se pueden devolver una o más columnas.

Si la rutina es una función de tabla, puede devolver cero o más filas con una o más columnas.

### Normas

- La ejecución de un método o una función SQL debe finalizar con una sentencia RETURN (SQLSTATE 42632).
- En una función de fila o tabla SQL que utiliza una *sentencia-dinámica-compuesta*, la única sentencia RETURN permitida es la que se encuentra al final de la sentencia compuesta (SQLSTATE 429BD).

### Notas

- Cuando se devuelve un valor de un procedimiento, el llamante puede acceder al valor:
  - utilizando la sentencia GET DIAGNOSTICS para recuperar el RETURN\_STATUS si el procedimiento SQL se ha llamado desde otro procedimiento SQL
  - utilizando el parámetro encontrado para el marcador de parámetro de valor devuelto en la sintaxis de la cláusula de escape CALL (?=CALL...) en una aplicación CLI
  - directamente del campo SQLERRD[0] de SQLCA, después de procesar la llamada (CALL) de un procedimiento SQL. Este campo sólo es válido si SQLCODE es cero o positivo (de lo contrario, suponga un valor de -1).

### Ejemplos

Utilice una sentencia RETURN para salir de un procedimiento almacenado SQL con un valor de estado de cero si la ejecución es satisfactoria y de -200 si no lo es.

```
BEGIN
...
GOTO FAIL
```

```

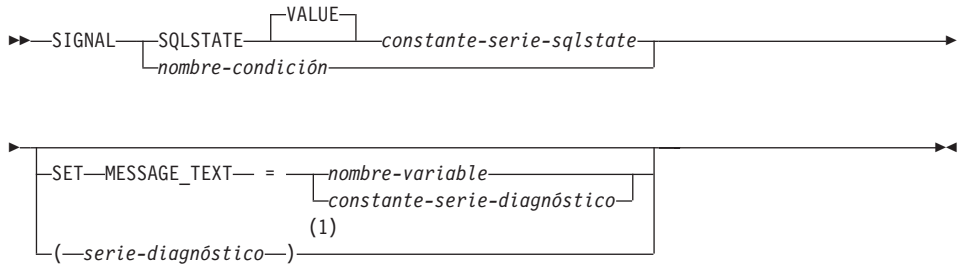
SUCCESS: RETURN 0
FAIL: RETURN -200
END
```

## Sentencia SIGNAL

### Sentencia SIGNAL

La sentencia SIGNAL se utiliza para transmitir una condición de error o de aviso. Hace que se devuelva un error o un aviso con el SQLSTATE especificado, junto con texto de mensaje opcional.

#### Sintaxis



#### Notas:

- 1 Esta opción sólo se proporciona en el ámbito de la sentencia CREATE TRIGGER por compatibilidad con las versiones anteriores de DB2.

#### Descripción

##### SQLSTATE VALUE *constante-serie-sqlstate*

La constante especificada de tipo serie representa un estado SQL (SQLSTATE). Debe ser una constante de tipo serie de caracteres con exactamente 5 caracteres que siguen las reglas para QLSTATES:

- Cada carácter debe pertenecer al conjunto de dígitos ('0' a '9') o de letras mayúsculas no acentuadas ('A' a 'Z').
- La clase SQLSTATE (primeros dos caracteres) no puede ser '00', pues esto representa una finalización satisfactoria.

En el contexto de una sentencia dinámica compuesta, un desencadenante, una función SQL o un método SQL, se deben aplicar también las reglas siguientes:

- La clase SQLSTATE (dos primeros caracteres) no puede ser '01' o '02', dado que éstos no son clases de error.
- Si la clase SQLSTATE empieza con los números '0' a '6' o las letras 'A' a 'H', la subclase (los tres últimos caracteres) debe empezar con una letra en el rango de 'T' a 'Z'.
- Si la clase SQLSTATE empieza con los números '7', '8', '9' o las letras 'I' a 'Z', la subclase puede ser cualquier carácter de '0' a '9' o de 'A' a 'Z'.

Si SQLSTATE no se ajusta a estas reglas, se produce un error (SQLSTATE 428B3).

*nombre-condición*

Especifica el nombre de la condición. El nombre de la condición debe ser exclusivo dentro del procedimiento y sólo puede estar referenciado dentro de la sentencia compuesta donde está declarado.

**SET MESSAGE\_TEXT=**

Especifica una serie de caracteres que describe el error o aviso. La serie de caracteres se coloca en el campo SQLERRMC de la SQLCA. Si la serie tiene más de 70 bytes de longitud, se truncará sin avisar de ello. Esta cláusula sólo puede especificarse si también está especificado un SQLSTATE o nombre-condición (SQLSTATE 42601).

*nombre-variable*

Identifica una variable SQL que se debe declarar dentro de la sentencia compuesta. La variable SQL debe estar definida con el tipo de datos CHAR o VARCHAR.

*constante-serie-diagnóstico*

Especifica una constante de tipo serie que contiene el texto del mensaje.

*serie-diagnóstico*

Una expresión con un tipo de CHAR o VARCHAR que devuelve una serie de caracteres de hasta 70 bytes para describir la condición de error. Si la serie tiene más de 70 bytes de longitud, se truncará. Esta opción sólo se proporciona en el ámbito de una sentencia CREATE TRIGGER por compatibilidad con las versiones anteriores de DB2. Se recomienda no utilizarla de forma continua.

**Notas**

- Si se emite una sentencia SIGNAL, el SQLCODE que se asigna es:
  - +438 si el SQLSTATE comienza con '01' o '02'
  - 438 en otro caso
- Si el SQLSTATE o la condición indica que se señala una excepción (clase SQLSTATE distinta de '01' o '02'):
  - Se maneja la excepción y se transfiere el control a un manejador, a condición de que exista un manejador en la misma sentencia compuesta (o una sentencia compuesta más externa) que la sentencia de señal y la sentencia compuesta contenga un manejador para el SQLSTATE, el nombre-condición o la SQLEXCEPTION especificados
  - Si la excepción no se puede manejar, se devuelve inmediatamente el control al final de la sentencia compuesta.
- Si el SQLSTATE o la condición indica que se señala un aviso (clase SQLSTATE '01') o una condición de no encontrado (not found) (clase SQLSTATE '02'):

## Sentencia SIGNAL

- Se maneja el aviso o la condición "not found" y se transfiere el control a un manejador, a condición de que exista un manejador en la misma sentencia compuesta (o una sentencia compuesta más externa) que la sentencia de señal y que la sentencia compuesta contenga un manejador para el SQLSTATE, el nombre-condición o el SQLWARNING especificados (si la clase SQLSTATE es '01') o NOT FOUND (si la clase SQLSTATE es '02');
- Si el aviso no se puede manejar, el proceso continúa con la sentencia siguiente.
- Los valores de SQLSTATE constan de un código de clase formado por dos caracteres, seguido de un código de subclase formado por tres caracteres. Los códigos de clase representan condiciones de ejecución satisfactoria o errónea.

Se puede utilizar un valor válido cualquiera de SQLSTATE en la sentencia SIGNAL. Sin embargo, es recomendable que el programador defina nuevos SQLSTATE basados en los rangos de valores reservados para las aplicaciones. Esto evita el uso involuntario de un valor de SQLSTATE que el gestor de bases de datos podría definir en un futuro release.
- Se pueden definir las clases de SQLSTATE que comienzan con los caracteres del '7' al '9', o de la 'I' a la 'Z'. Dentro de estas clases, se puede definir cualquier subclase.
- Las clases de SQLSTATE que comienzan con los caracteres del '0' al '6', o de la 'A' a la 'H' están reservadas para el gestor de bases de datos. Dentro de estas clases, las subclases que comienzan con los caracteres del '0' a la 'H' están reservadas para el gestor de bases de datos. Se pueden definir subclases que empiecen con los caracteres 'I' a 'Z'.

## Ejemplos

En el ejemplo siguiente, se utiliza un procedimiento SQL para un sistema de gestión de pedidos que señala un error de aplicación cuando la aplicación no reconoce un número de cliente. La tabla ORDERS incluye una clave foránea para la tabla CUSTOMER, lo cual hace necesario que exista el número de cliente (CUSTNO) para poder insertar un pedido.

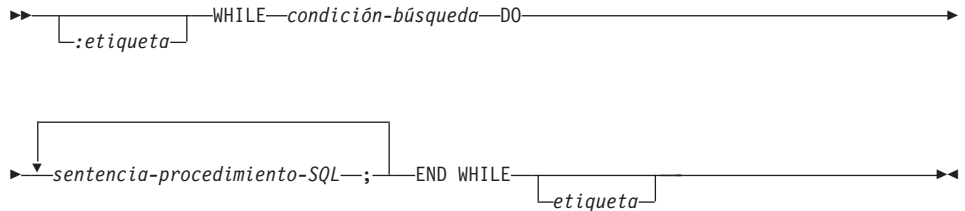
```
CREATE PROCEDURE SUBMIT_ORDER
(IN ONUM INTEGER, IN CNUM INTEGER,
 IN PNUM INTEGER, IN QNUM INTEGER)
SPECIFIC SUBMIT_ORDER
MODIFIES SQL DATA
LANGUAGE SQL
BEGIN
 DECLARE EXIT HANDLER FOR SQLSTATE VALUE '23503'
 SIGNAL SQLSTATE '75002'
 SET MESSAGE_TEXT = 'Customer number is not known!';
 INSERT INTO ORDERS (ORDERNO, CUSTNO, PARTNO, QUANTITY)
 VALUES (ONUM, CNUM, PNUM, QNUM);
END
```



## Sentencia WHILE

La sentencia WHILE repite la ejecución de una sentencia o grupo de sentencias mientras sea verdadera una condición especificada.

### Sintaxis



### Descripción

#### *etiqueta*

Especifica la etiqueta de la sentencia WHILE. Si se especifica la etiqueta inicial, esa etiqueta puede especificarse en sentencias LEAVE e ITERATE. Si se especifica la etiqueta final, ésta debe ser igual que la etiqueta inicial.

#### *condición-búsqueda*

Especifica una condición que se evalúa antes de cada ejecución del bucle. Si la condición es verdadera, se procesan las sentencias de procedimiento SQL incluidas en el bucle.

#### *sentencia-procedimiento-SQL*

Especifica la sentencia o sentencias de SQL del bucle que se deben ejecutar.

### Ejemplos

Este ejemplo utiliza una sentencia WHILE para ejecutar un proceso iterativo mediante sentencias FETCH y SET. Mientras el valor de la variable SQL *v\_counter* es menor que la mitad del número de empleados del departamento identificado por el parámetro de entrada *deptNumber*, la sentencia WHILE continúa ejecutando las sentencias FETCH y SET. Cuando la condición deja de ser verdadera, el flujo de control sale de la sentencia WHILE y cierra el cursor.

```
CREATE PROCEDURE DEPT_MEDIAN
(IN deptNumber SMALLINT, OUT medianSalary DOUBLE)
LANGUAGE SQL
BEGIN
 DECLARE v_numRecords INTEGER DEFAULT 1;
 DECLARE v_counter INTEGER DEFAULT 0;
 DECLARE c1 CURSOR FOR
 SELECT CAST(salary AS DOUBLE)
 FROM staff
 WHERE DEPT = deptNumber
 ORDER BY salary;
 DECLARE EXIT HANDLER FOR NOT FOUND
```

## Sentencia WHILE

```
 SET medianSalary = 6666;
SET medianSalary = 0;
SELECT COUNT(*) INTO v_numRecords
 FROM staff
 WHERE DEPT = deptNumber;
OPEN c1;
WHILE v_counter < (v_numRecords / 2 + 1) DO
 FETCH c1 INTO medianSalary;
 SET v_counter = v_counter + 1;
END WHILE;
CLOSE c1;
END
```

---

## Apéndice A. Límites de SQL

Las tablas siguientes describen algunos límites de SQL. Si el programador se ajusta al caso más restrictivo le servirá de ayuda para diseñar programas de aplicación que sean fácilmente portables.

Tabla 31. Límites de longitud del identificador

|    | Descripción                                                                                    | Límite en bytes |
|----|------------------------------------------------------------------------------------------------|-----------------|
| 1  | El nombre de autorización más largo (sólo puede ser de caracteres de un solo byte)             | 30              |
| 2  | Nombre de restricción más largo                                                                | 18              |
| 3  | Nombre de correlación más largo                                                                | 128             |
| 4  | Nombre de condición más largo                                                                  | 64              |
| 5  | Nombre de cursor más largo                                                                     | 18              |
| 6  | Nombre más largo de columna de fuente de datos                                                 | 128             |
| 7  | Nombre más largo de índice de fuente de datos                                                  | 128             |
| 8  | Nombre más largo de fuente de datos                                                            | 128             |
| 9  | Nombre más largo de tabla de fuente de datos ( <i>nombre-tabla-remota</i> )                    | 128             |
| 10 | Nombre de programa externo más largo                                                           | 8               |
| 11 | Identificador del lenguaje principal más largo <sup>a</sup>                                    | 255             |
| 12 | Identificador más largo de un usuario de fuente de datos ( <i>nombre-autorización-remota</i> ) | 30              |
| 13 | Nombre de etiqueta más largo                                                                   | 64              |
| 14 | Nombre de método más largo                                                                     | 18              |
| 15 | Nombre de parámetro más largo <sup>b</sup>                                                     | 128             |
| 16 | Palabra clave más larga para acceder a una fuente de datos                                     | 32              |
| 17 | Nombre de punto de salvar más largo                                                            | 128             |
| 18 | Nombre de esquema más largo <sup>c</sup>                                                       | 30              |
| 19 | Nombre de servidor (seudónimo de base de datos) más largo                                      | 8               |
| 20 | Nombre de variable SQL más largo                                                               | 64              |
| 21 | Nombre de sentencia más largo                                                                  | 18              |
| 22 | Nombre de grupo de transformación más largo                                                    | 18              |

## Límites de SQL

Tabla 31. Límites de longitud del identificador (continuación)

|    | Descripción                                                                                                                                                                                                                       | Límite en bytes |
|----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| 23 | Nombre de columna no calificada más largo                                                                                                                                                                                         | 30              |
| 24 | Nombre de paquete no calificado más largo                                                                                                                                                                                         | 8               |
| 25 | Nombre más largo del tipo definido por el usuario no calificado, función definida por el usuario, agrupación de almacenamientos intermedios, espacio de tablas, grupo de nodos, desencadenante, índice o especificación de índice | 18              |
| 26 | Nombre más largo de tabla no calificado, nombre de vista, procedimiento almacenado, apodo o seudónimo                                                                                                                             | 128             |
| 27 | Nombre más largo de reiniciador                                                                                                                                                                                                   | 128             |

### Notas:

- a Los compiladores de lenguaje principal individuales pueden aplicar límites más restrictivos a los nombres de variable.
- b En un procedimiento SQL los nombres de parámetros tienen una longitud máxima de 64 bytes.
- c El nombre de esquema para un tipo definido por el usuario está limitado a 8 bytes.

Tabla 32. Límites numéricos

|    | Descripción                  | Límite                     |
|----|------------------------------|----------------------------|
| 1  | Valor INTEGER más pequeño    | -2 147 483 648             |
| 2  | Valor INTEGER máximo         | +2 147 483 647             |
| 3  | Valor BIGINT más pequeño     | -9 223 372 036 854 775 808 |
| 4  | Valor BIGINT máximo          | +9 223 372 036 854 775 807 |
| 5  | Valor SMALLINT mínimo        | -32 768                    |
| 6  | Valor SMALLINT máximo        | +32 767                    |
| 7  | Precisión decimal máxima     | 31                         |
| 8  | Valor DOUBLE mínimo          | -1,79769E+308              |
| 9  | Valor DOUBLE máximo          | +1,79769E+308              |
| 10 | Valor DOUBLE positivo mínimo | +2,225E-307                |
| 11 | Valor DOUBLE negativo máximo | -2,225E-307                |
| 12 | Valor REAL mínimo            | -3,402E+38                 |
| 13 | Valor REAL máximo            | +3,402E+38                 |
| 14 | Valor REAL positivo mínimo   | +1,175E-37                 |

Tabla 32. Límites numéricos (continuación)

|    | Descripción                | Límite     |
|----|----------------------------|------------|
| 15 | Valor REAL negativo máximo | -1,175E-37 |

Tabla 33. Límites de series

|    | Descripción                                                                               | Límite        |
|----|-------------------------------------------------------------------------------------------|---------------|
| 1  | Longitud máxima de CHAR (en bytes)                                                        | 254           |
| 2  | Longitud máxima de VARCHAR (en bytes)                                                     | 32 672        |
| 3  | Longitud máxima de LONG VARCHAR (en bytes)                                                | 32 700        |
| 4  | Longitud máxima de CLOB (en bytes)                                                        | 2 147 483 647 |
| 5  | Longitud máxima de GRAPHIC (en caracteres)                                                | 127           |
| 6  | Longitud máxima de VARGRAPHIC (en caracteres)                                             | 16 336        |
| 7  | Longitud máxima de LONG VARGRAPHIC (en caracteres)                                        | 16 350        |
| 8  | Longitud máxima de DBCLOB (en caracteres)                                                 | 1 073 741 823 |
| 9  | Longitud máxima de BLOB (en bytes)                                                        | 2 147 483 647 |
| 10 | Longitud máxima de constante de caracteres                                                | 32 672        |
| 11 | Longitud máxima de constante gráfica                                                      | 16 336        |
| 12 | Longitud máxima de series de caracteres concatenadas                                      | 2 147 483 647 |
| 13 | Longitud máxima de series gráficas concatenadas                                           | 1 073 741 823 |
| 14 | Longitud máxima de series binarias concatenadas                                           | 2 147 483 647 |
| 15 | Número máximo de dígitos de constante hexadecimal                                         | 16 336        |
| 16 | Tamaño máximo de un comentario del catálogo (en bytes)                                    | 254           |
| 17 | Instancia más larga de un objeto de una columna de tipo estructurado durante la ejecución | 1 GB          |

Tabla 34. Límites de fecha y hora

|   | Descripción       | Límite     |
|---|-------------------|------------|
| 1 | Valor DATE mínimo | 0001-01-01 |
| 2 | Valor DATE máximo | 9999-12-31 |

## Límites de SQL

Tabla 34. Límites de fecha y hora (continuación)

|   | Descripción            | Límite                     |
|---|------------------------|----------------------------|
| 3 | Valor TIME mínimo      | 00:00:00                   |
| 4 | Valor TIME máximo      | 24:00:00                   |
| 5 | Valor TIMESTAMP mínimo | 0001-01-01-00.00.00.000000 |
| 6 | Valor TIMESTAMP máximo | 9999-12-31-24.00.00.000000 |

Tabla 35. Límites del gestor de bases de datos

|    | Descripción                                                                                                              | Límite                  |
|----|--------------------------------------------------------------------------------------------------------------------------|-------------------------|
| 1  | Máximo número de columnas en una tabla <sup>g</sup>                                                                      | 1 012                   |
| 2  | Máximo número de columnas en una vista <sup>a</sup>                                                                      | 5 000                   |
| 3  | Longitud máxima de una fila incluyendo toda la actividad general <sup>b g</sup>                                          | 32 677                  |
| 4  | Tamaño máximo de una tabla por partición(en gigabytes) <sup>c g</sup>                                                    | 512                     |
| 5  | Tamaño máximo de un índice por partición(en gigabytes)                                                                   | 512                     |
| 6  | Número máximo de filas en una tabla por partición                                                                        | 4 x 10 <sup>9</sup>     |
| 7  | Clave de índice más larga incluyendo toda la actividad general (en bytes)                                                | 1 024                   |
| 8  | Número máximo de columnas en una clave de índice                                                                         | 16                      |
| 9  | Número máximo de índices en una tabla                                                                                    | 32 767 o almacenamiento |
| 10 | Número máximo de tablas de referencia en una sentencia de SQL o una vista                                                | almacenamiento          |
| 11 | Número máximo de declaraciones de variables del lenguaje principal en un programa precompilado <sup>c</sup>              | almacenamiento          |
| 12 | Número máximo de referencias de variables del lenguaje principal en una sentencia de SQL                                 | 32 767                  |
| 13 | Valor más largo de variable del lenguaje principal utilizado para operaciones de inserción o de actualización (en bytes) | 2 147 483 647           |
| 14 | Sentencia de SQL más larga (en bytes)                                                                                    | 65 535                  |
| 15 | Número máximo de elementos en una lista de selección <sup>g</sup>                                                        | 1 012                   |

Tabla 35. Límites del gestor de bases de datos (continuación)

|    | Descripción                                                                                                          | Límite         |
|----|----------------------------------------------------------------------------------------------------------------------|----------------|
| 16 | Número máximo de predicados en una cláusula WHERE o HAVING                                                           | almacenamiento |
| 17 | Número máximo de columnas en una cláusula GROUP BY <sup>g</sup>                                                      | 1 012          |
| 18 | Longitud total máxima de las columnas en una cláusula GROUP BY (en bytes) <sup>g</sup>                               | 32 677         |
| 19 | Número máximo de columnas en una cláusula ORDER BY <sup>g</sup>                                                      | 1 012          |
| 20 | Longitud total máxima de las columnas en una cláusula ORDER BY (en bytes) <sup>g</sup>                               | 32 677         |
| 21 | Tamaño máximo de una SQLDA (en bytes)                                                                                | almacenamiento |
| 22 | Número máximo de sentencias preparadas                                                                               | almacenamiento |
| 23 | Número máximo de cursores declarados en un programa                                                                  | almacenamiento |
| 24 | Número máximo de cursores abiertos a la vez                                                                          | almacenamiento |
| 25 | Número máximo de tablas en un espacio de tablas SMS                                                                  | 65 534         |
| 26 | Número máximo de restricciones en una tabla                                                                          | almacenamiento |
| 27 | Nivel máximo de anidamiento de subconsultas                                                                          | almacenamiento |
| 28 | Número máximo de subconsultas en una sola sentencia                                                                  | almacenamiento |
| 29 | Número máximo de valores en una sentencia INSERT <sup>g</sup>                                                        | 1 012          |
| 30 | Número máximo de cláusulas SET en una sola sentencia UPDATE <sup>g</sup>                                             | 1 012          |
| 31 | Número máximo de columnas en una restricción UNIQUE (soportado a través de un índice UNIQUE)                         | 16             |
| 32 | Longitud máxima combinada de las columnas de una restricción UNIQUE (soportada mediante un índice UNIQUE) (en bytes) | 1 024          |
| 33 | Número máximo de columnas de referencia en una clave foránea                                                         | 16             |
| 34 | Longitud máxima combinada de columnas de referencia en una clave foránea (en bytes)                                  | 1 024          |
| 35 | Longitud máxima de una especificación de restricción de comprobación (en bytes)                                      | 65 535         |

## Límites de SQL

Tabla 35. Límites del gestor de bases de datos (continuación)

|    | Descripción                                                                    | Límite               |
|----|--------------------------------------------------------------------------------|----------------------|
| 36 | Número máximo de columnas en una clave de particionamiento <sup>e</sup>        | 500                  |
| 37 | Número máximo de filas cambiadas en una unidad de trabajo                      | almacenamiento       |
| 38 | Número máximo de paquetes                                                      | almacenamiento       |
| 39 | Número máximo de constantes en una sentencia                                   | almacenamiento       |
| 40 | Número máximo de usuarios simultáneos del servidor <sup>d</sup>                | 64 000               |
| 41 | Número máximo de parámetros en un procedimiento almacenado                     | 32 767               |
| 42 | Número máximo de parámetros en una función definida por el usuario             | 90                   |
| 43 | Profundidad máxima en tiempo de ejecución de desencadenantes en cascada        | 16                   |
| 44 | Número máximo de supervisores de sucesos activos simultáneamente               | 32                   |
| 45 | Tamaño máximo de un espacio de tablas DMS normal (en gigabytes) <sup>c g</sup> | 512                  |
| 46 | Tamaño máximo de un espacio de tablas DMS largo (en terabytes) <sup>c</sup>    | 2                    |
| 47 | Tamaño máximo de un espacio de tablas DMS temporal (en terabytes) <sup>c</sup> | 2                    |
| 48 | Número máximo de bases de datos por instancia en uso simultáneo                | 256                  |
| 49 | Número máximo de usuarios simultáneos por instancia                            | 64 000               |
| 50 | Número máximo de aplicaciones simultáneas por base de datos                    | 1 000                |
| 51 | Profundidad máxima de desencadenantes en cascada                               | 16                   |
| 52 | Número de partición máximo                                                     | 999                  |
| 53 | Número máximo de objetos de tabla en un espacio de tablas DMS <sup>f</sup>     | 51 000               |
| 54 | Parte más larga de la clave de índice variable (en bytes) <sup>h</sup>         | 255 o almacenamiento |



Tabla 35. Límites del gestor de bases de datos (continuación)

|    | Descripción                                                                                                     | Límite        |
|----|-----------------------------------------------------------------------------------------------------------------|---------------|
| 55 | Número máximo de columnas en una vista o tabla de fuente de datos a la que se hace referencia mediante un apodo | 5 000         |
| 56 | Valor máximo de NPAGES en una agrupación de almacenamientos intermedios para emisiones de 32 bits               | 524 288       |
| 57 | Valor máximo de NPAGES en una agrupación de almacenamientos intermedios para emisiones de 64 bits               | 2 147 483 647 |
| 58 | Número máximo de niveles de anidamiento para procedimientos almacenados                                         | 16            |
| 59 | Número máximo de espacios de tablas en una base de datos                                                        | 4096          |
| 60 | Número máximo de atributos en un tipo estructurado                                                              | 4082          |

**Notas:**

- a** Este máximo puede conseguirse utilizando una unión en la sentencia CREATE VIEW. La selección de dicha vista está sujeta al límite del número máximo de elementos de una lista de selección.
- b** Los datos reales para las columnas BLOB, CLOB, LONG VARCHAR, DBCLOB y LONG VARGRAPHIC no se incluyen en esta cuenta. Sin embargo, la información acerca de la ubicación de los datos ocupa espacio en la fila.
- c** Los números mostrados son límites y aproximaciones arquitectónicos. En la práctica los límites pueden ser menores.
- d** El valor real será el valor del parámetro de configuración MAXAGENTS. Consulte el manual *Administration Guide* para obtener información acerca de MAXAGENTS.
- e** Es un límite de arquitectura. El límite en la mayoría de las columnas de una clave de índice debe utilizarse como un límite práctico.
- f** Los objetos de tabla incluyen datos, índices, columnas LONG VARCHAR/VARGRAPHIC y columnas LOB. Los objetos de tabla que están en el mismo espacio de tablas que los datos de tabla no cuentan como adicionales respecto al límite. No obstante, cada objeto de tabla que está en un espacio de tablas diferente de los datos de tabla representa uno respecto al límite para cada tipo de objeto de tabla por tabla en el espacio de tablas en que reside el objeto de tabla.
- g** Para conocer los valores específicos de tamaño de página, consulte la Tabla 36 en la página 1260.
- h** Para que una parte de clave de índice variable tenga más de 255 bytes, la variable de registro DB2\_INDEX\_2BYTEVARLEN debe establecerse en "ON".

## Límites de SQL

Tabla 36. Límites específicos de tamaño de página del gestor de bases de datos

|    | Descripción                                                               | Límite de tamaño de página de 4K | Límite de tamaño de página de 8K | Límite de tamaño de página de 16K | Límite de tamaño de página de 32K |
|----|---------------------------------------------------------------------------|----------------------------------|----------------------------------|-----------------------------------|-----------------------------------|
| 1  | Máximo número de columnas en una tabla                                    | 500                              | 1 012                            | 1 012                             | 1 012                             |
| 3  | Longitud máxima de una fila incluyendo toda la actividad general          | 4 005                            | 8 101                            | 16 293                            | 32 677                            |
| 4  | Tamaño máximo de una tabla por partición (en gigabytes)                   | 64                               | 128                              | 256                               | 512                               |
| 5  | Tamaño máximo de un índice por partición(en gigabytes)                    | 64                               | 128                              | 256                               | 512                               |
| 15 | Número máximo de elementos en una lista de selección                      | 500                              | 1 012                            | 1 012                             | 1 012                             |
| 17 | Número máximo de columnas en una cláusula GROUP BY                        | 500                              | 1 012                            | 1 012                             | 1 012                             |
| 18 | Longitud total máxima de las columnas en una cláusula GROUP BY (en bytes) | 4 005                            | 8 101                            | 16 293                            | 32 677                            |
| 19 | Número máximo de columnas en una cláusula ORDER BY                        | 500                              | 1 012                            | 1 012                             | 1 012                             |
| 20 | Longitud total máxima de las columnas en una cláusula ORDER BY (en bytes) | 4 005                            | 8 101                            | 16 293                            | 32 677                            |
| 29 | Número máximo de valores en una sentencia INSERT                          | 500                              | 1 012                            | 1 012                             | 1 012                             |
| 30 | Número máximo de cláusulas SET en una sola sentencia UPDATE               | 500                              | 1 012                            | 1 012                             | 1 012                             |
| 45 | Tamaño máximo de un espacio de tablas DMS normal (en gigabytes)           | 64                               | 128                              | 256                               | 512                               |

---

## Apéndice B. Comunicaciones SQL (SQLCA)

Una SQLCA es un conjunto de variables que se actualiza al final de la ejecución de cada sentencia de SQL. Un programa que contiene sentencias de SQL ejecutables (excepto DECLARE, INCLUDE y WHENEVER) y se precompila con la opción LANGLEVEL SAA1 (el valor por omisión) o MIA debe proporcionar exactamente una SQLCA, aunque es posible que existan más de una SQLCA por paso en una aplicación de múltiples pasos.

Cuando se precompila un programa con la opción LANGLEVEL SQL92E, puede declararse una variable SQLCODE o SQLSTATE en la sección de declaración SQL o se puede declarar una variable SQLCODE en algún otro lugar del programa.

No se debe proporcionar ninguna SQLCA cuando se utiliza LANGLEVEL SQL92E. La sentencia de SQL INCLUDE puede utilizarse para proporcionar la declaración de la SQLCA en todos los lenguajes excepto en REXX. La SQLCA se proporciona automáticamente en REXX.

---

### Visualización de SQLCA interactivamente

Para visualizar la SQLCA después de cada mandato que utilice en el procesador de línea de mandatos, utilice el mandato **db2 -a**. La SQLCA se proporciona como parte de la salida para los mandatos posteriores. La SQLCA también se vuelca en el archivo db2diag.log file.

---

### Descripciones de los campos de la SQLCA

Tabla 37. Campos de SQLCA

| Nombre <sup>111</sup> | Tipo de datos | Valores de campos                                                                                                                                                                                          |
|-----------------------|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| sqlcaid               | CHAR(8)       | Una indicación visual para los vuelcos de almacenamiento que contienen la 'SQLCA'. El sexto byte es 'L' si el número de línea devuelto procede del análisis sintáctico del cuerpo de un procedimiento SQL. |
| sqlcab                | INTEGER       | Contiene la longitud de la SQLCA, 136.                                                                                                                                                                     |

---

111. Los nombres de campos mostrados son los que están presentes en una SQLCA obtenida mediante una sentencia INCLUDE.

## SQLCA

Tabla 37. Campos de SQLCA (continuación)

| Nombre <sup>111</sup> | Tipo de datos | Valores de campos                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-----------------------|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| sqlcode               | INTEGER       | <p>Contiene el código de retorno SQL. Para ver los significados específicos de los códigos de retorno, consulte la sección de mensajes del manual <i>Consulta de mensajes</i>.</p> <p><b>Código Significado</b></p> <p><b>0</b> Ejecución satisfactoria (aunque pueden haberse establecido uno o varios indicadores SQLWARN).</p> <p><b>positivo</b><br/>Ejecución satisfactoria, pero con una condición de aviso.</p> <p><b>negativo</b><br/>Condición de error.</p>                                     |
| sqlerrml              | SMALLINT      | Indicador de longitud para <i>sqlerrmc</i> , en el rango de 0 a 70. 0 significa que el valor de <i>sqlerrmc</i> no es relevante.                                                                                                                                                                                                                                                                                                                                                                          |
| sqlerrmc              | VARCHAR (70)  | <p>Contiene uno o más símbolos, separados por X'FF', que sustituyen a variables en las descripciones de las condiciones de error.</p> <p>Este campo también se utiliza cuando se establece una conexión satisfactoria.</p> <p>Cuando se emite una sentencia de SQL compuesta NOT ATOMIC, puede contener información acerca de un máximo de 7 errores.</p> <p>Para ver los significados específicos de los códigos de retorno, consulte la sección de mensajes del manual <i>Consulta de mensajes</i>.</p> |
| sqlerrp               | CHAR(8)       | <p>Empieza con un identificador de tres letras que indica el producto, seguido de cinco dígitos indicando la versión, release y nivel de modificación del producto. Por ejemplo, SQL07010 significa DB2 Universal Database Versión 7, Release 1, Nivel de modificación 0.</p> <p>Si SQLCODE indica una condición de error, entonces este campo identifica el módulo que ha devuelto el error.</p> <p>Este campo también se utiliza cuando se completa una conexión satisfactoria.</p>                     |
| sqlerrd               | ARRAY         | Seis variables INTEGER que proporcionan información de diagnóstico. Generalmente, estos valores están vacíos si no hay errores, excepto sqlerrd(6) de una base de datos particionada.                                                                                                                                                                                                                                                                                                                     |

Tabla 37. Campos de SQLCA (continuación)

| Nombre <sup>111</sup> | Tipo de datos | Valores de campos                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-----------------------|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| sqlerrd(1)            | INTEGER       | <p>Si se invoca la conexión y es satisfactoria, contiene la diferencia máxima esperada en la longitud de los datos de caracteres mixtos (tipos de datos CHAR) cuando se convierten a la página de códigos de la base de datos de la página de códigos de la aplicación. Un valor de 0 ó 1 indica sin expansión; un valor mayor que 1 indica una posible expansión en longitud; un valor negativo indica una posible contracción. <sup>a</sup></p> <p>Cuando un procedimiento SQL termina satisfactoriamente su ejecución, este campo contiene el valor del estado de retorno del procedimiento SQL.</p>                                                                                                                             |
| sqlerrd(2)            | INTEGER       | <p>Si se invoca una conexión y es satisfactoria, contiene la diferencia máxima de longitud esperada para los datos de caracteres mixtos (tipos de datos CHAR) cuando se convierten a la página de códigos de la aplicación a partir de la página de códigos de la base de datos. Un valor de 0 ó 1 indica sin expansión; un valor mayor que 1 indica una posible expansión en longitud; un valor negativo indica una posible contracción. <sup>a</sup> Si la SQLCA es el resultado de una sentencia de SQL compuesta NOT ATOMIC que ha encontrado uno o varios errores, el valor se establece en el número de sentencias que han fallado.</p>                                                                                       |
| sqlerrd(3)            | INTEGER       | <p>Si se invoca PREPARE y es satisfactoria, contiene una estimación del número de filas que se devolverán. Después de INSERT, UPDATE y DELETE, contiene el número real de filas afectadas. Si se invoca el SQL compuesto, contiene una acumulación de todas las filas de subsentencia. Si se invoca CONNECT, contiene 1 si la base de datos se puede actualizar; 2 si la base de datos es de sólo lectura.</p> <p>Si se invoca CREATE PROCEDURE para un procedimiento SQL y se detecta un error durante el análisis sintáctico del cuerpo del procedimiento SQL, este campo contiene el número de la línea donde se encontró el error. El sexto byte de sqlcaid debe ser 'L' para que este valor sea un número de línea válido.</p> |
| sqlerrd(4)            | INTEGER       | <p>Si se invoca PREPARE y es satisfactoria, contiene una estimación del coste relativo de los recursos necesarios para procesar la sentencia. Si se invoca el SQL compuesto, contiene una cuenta del número de subsentencias satisfactorias. Si se invoca CONNECT, contiene 0 para una confirmación en una fase de un cliente de nivel inferior; 1 para una confirmación de una fase; 2 para confirmación de sólo lectura de una fase; y 3 para una confirmación de dos fases.</p>                                                                                                                                                                                                                                                  |

## SQLCA

Tabla 37. Campos de SQLCA (continuación)

| Nombre <sup>111</sup> | Tipo de datos | Valores de campos                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-----------------------|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| sqlerrd(5)            | INTEGER       | <p>Contiene el número total de filas suprimidas, insertadas o actualizadas como resultado de:</p> <ul style="list-style-type: none"><li>• La imposición de las restricciones después de una operación de supresión satisfactoria</li><li>• El proceso de sentencias de SQL activadas por desencadenantes activados.</li></ul> <p>Si se invoca el SQL compuesto, contiene una acumulación del número de dichas filas para todas las subsentencias. En algunos casos cuando se encuentra un error, este campo contiene un valor negativo que es un puntero de un error interno. Si se invoca CONNECT, contiene el valor de tipo de autenticación 0 para una autenticación de servidor; 1 para la autenticación de cliente; 2 para la autenticación mediante la utilización de DB2 Connect; 3 para la autenticación de los servicios de seguridad DCE; 255 para la autenticación no especificada.</p> |
| sqlerrd(6)            | INTEGER       | <p>Para una base de datos particionada, contiene el número de partición de la partición que ha encontrado el error o aviso. Si no se han encontrado errores ni avisos, este campo contiene el número de partición del nodo coordinador. El número de este campo es igual al especificado para la partición del archivo db2nodes.cfg.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| sqlwarn               | Matriz        | <p>Un conjunto de indicadores de aviso, que contiene cada uno un blanco o W. Si se invoca el SQL compuesto, contiene una acumulación de indicadores de aviso establecidos para todas las subsentencias.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| sqlwarn0              | CHAR(1)       | <p>Blanco si todos los demás indicadores están en blanco; contiene W si como mínimo otro indicador no está en blanco.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| sqlwarn1              | CHAR(1)       | <p>Contiene una "W" si el valor de una columna de serie se ha truncado cuando se ha asignado a una variable del lenguaje principal. Contiene una "N" si el terminador nulo se ha truncado.</p> <p>Contiene una "A" si la operación CONNECT o ATTACH se realiza satisfactoriamente y el ID de autorización de la conexión tiene más de 8 bytes.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| sqlwarn2              | CHAR(1)       | <p>Contiene W si los valores nulos se han eliminado del argumento de una función. <sup>b</sup></p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| sqlwarn3              | CHAR(1)       | <p>Contiene W si el número de columnas no es igual al número de variables del lenguaje principal.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| sqlwarn4              | CHAR(1)       | <p>Contiene W si una sentencia UPDATE o DELETE preparada no incluye una cláusula WHERE.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

Tabla 37. Campos de SQLCA (continuación)

| Nombre <sup>111</sup> | Tipo de datos | Valores de campos                                                                                                                                                                                              |
|-----------------------|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| sqlwarn5              | CHAR(1)       | Reservado para su utilización en el futuro.                                                                                                                                                                    |
| sqlwarn6              | CHAR(1)       | Contiene W si se ha ajustado el resultado del cálculo de una fecha para evitar una fecha imposible.                                                                                                            |
| sqlwarn7              | CHAR(1)       | Reservado para su utilización en el futuro.<br><br>Si se invoca CONNECT y se ejecuta satisfactoriamente, contiene una "E" si el parámetro de configuración DYN_QUERY_MGMT de la base de datos está habilitado. |
| sqlwarn8              | CHAR(1)       | Contiene W si el carácter que no se ha podido convertir se ha sustituido por un carácter de sustitución.                                                                                                       |
| sqlwarn9              | CHAR(1)       | Contiene W si se han pasado por alto las expresiones aritméticas que contienen errores durante el proceso de función de columna.                                                                               |
| sqlwarn10             | CHAR(1)       | Contiene W si ha habido un error de conversión al convertir un valor de datos de caracteres de uno de los campos de la SQLCA.                                                                                  |
| sqlstate              | CHAR(5)       | Un código de retorno que indica el resultado de la sentencia de SQL ejecutada más recientemente.                                                                                                               |

**Nota:**

- a Consulte los detalles en la sección "Character Conversion Expansion Factor" del capítulo "Programming in Complex Environments" del manual *Application Development Guide*.
- b Es posible que algunas funciones no establezcan SQLWARN2 en W aunque se hayan eliminado los valores nulos porque el resultado no dependía de la eliminación de dichos valores.

**Orden del informe de errores**

El orden del informe de errores es el siguiente:

1. Las condiciones de error graves siempre se informan. Cuando se informa de un error grave, no hay ninguna adición a la SQLCA.
2. Si no se produce ningún error grave, un error de punto muerto tiene prioridad sobre los demás errores.
3. En el resto de errores, se devuelve la SQLCA para el primer código SQL negativo.
4. Si no se detecta ningún código SQL negativo, se devuelve la SQLCA para el primer aviso (es decir, código SQL positivo).

En DB2 Enterprise - Extended Edition, se produce una excepción a esta regla si se emite una operación de manipulación de datos en una tabla que está vacía en una partición, pero que tiene datos en otros nodos. Sólo se devuelve el SQLCODE +100 a la aplicación si los agentes de todas las

## SQLCA

particiones devuelven SQL0100W, porque la tabla está vacía en todas las particiones o porque no hay más filas que cumplan la cláusula WHERE de una sentencia UPDATE.

---

### Utilización de la SQLCA por DB2 Enterprise - Extended Edition

En DB2 Universal Database Enterprise - Extended Edition, una sentencia de SQL puede ejecutarse por un número de agentes de distintas particiones y cada agente puede devolver una SQLCA distinta para diferentes errores o avisos. El agente coordinador también tiene su propia SQLCA.

Para proporcionar una vista coherente para las aplicaciones, todos los valores de SQLCA se fusionan en una estructura y los campos de SQLCA indican cuentas globales. Por ejemplo:

- Para todos los errores y avisos, el campo *sqlwarn* contiene los distintivos de aviso recibidos de todos los agentes.
- Los valores de los campos *sqlerrd* que indican cuentas de fila son acumulaciones de todos los agentes.

Observe que es posible que no se devuelva SQLSTATE 09000 en todos los casos en que se produzca un error al procesar una sentencia de SQL activada.



---

## Apéndice C. Área de descriptores SQL (SQLDA)

Una SQLDA es un conjunto de variables que son necesarias para la ejecución de la sentencia de SQL DESCRIBE. Las variables SQLDA son opciones que las sentencias PREPARE, OPEN, FETCH, EXECUTE y CALL pueden utilizar. Una SQLDA se comunica con el SQL dinámico; puede utilizarse en una sentencia DESCRIBE, modificarse con las direcciones de las variables del lenguaje principal y después volverse a utilizar en una sentencia FETCH.

Se da soporte a SQLDA para todos los lenguajes, pero sólo se proporcionan declaraciones predefinidas para C, REXX, FORTRAN y COBOL. En REXX, la SQLDA es algo diferente que en los demás lenguajes; para obtener información sobre la utilización de las SQLDA en REXX consulte el manual *Application Development Guide*.

El significado de la información de una SQLDA depende de su utilización. En PREPARE y DESCRIBE, una SQLDA proporciona información para un programa de aplicación acerca de una sentencia preparada. En OPEN, EXECUTE, FETCH y CALL, una SQLDA describe las variables del lenguaje principal.

En DESCRIBE y PREPARE, si cualquiera de las columnas que se describen es de tipo LOB<sup>112</sup>, de tipo de referencia o un tipo definido por el usuario, el número de entradas de SQLVAR para la SQLDA completa se doblará. Por ejemplo:

- Cuando se describe una tabla con 3 columnas VARCHAR y 1 columna INTEGER, habrán 4 entradas SQLVAR
- Cuando se describe una tabla con 2 columnas VARCHAR, 1 columna CLOB y 1 columna de enteros, habrán 8 entradas SQLVAR

En EXECUTE, FETCH, OPEN y CALL, si cualquiera de las variables que se describe es un tipo LOB<sup>112</sup> o un tipo estructurado, se debe doblar el número de entradas de SQLVAR para la SQLDA completa.<sup>113</sup>

---

112. Los localizadores de LOB y las variables de referencia a archivos no necesitan SQLDA dobles.

113. Estos casos no son aplicables a los tipos diferenciados ni a los tipos de referencia, pues la base de datos no necesita la información adicional que proporcionan las entradas dobles.

---

### Descripciones de los campos

Una SQLDA consta de cuatro variables seguidas de un número arbitrario de ocurrencias de una secuencia de variables llamadas en conjunto SQLVAR. En OPEN, FETCH, EXECUTE y CALL cada ocurrencia de SQLVAR describe una variable del lenguaje principal. En DESCRIBE y PREPARE, cada ocurrencia de SQLVAR describe una columna de la tabla resultante. Hay dos tipos de entradas SQLVAR:

1. **SQLVAR base:** Estas entradas siempre están presentes. Contienen la información base acerca de la columna o variable del lenguaje principal como, por ejemplo, el código del tipo de datos, el atributo de longitud, el nombre de columna, la dirección de la variable del lenguaje principal y la dirección de la variable indicadora.
2. **SQLVAR secundarias:** Estas entradas sólo están presentes si el número de entradas SQLVAR se dobla por las reglas indicadas arriba. Para los tipos definidos por el usuario (diferenciado o estructurado), contienen el nombre del tipo definido por el usuario. Para los tipos de referencia, contienen ese tipo de destino de la referencia. Para los LOB, contienen el atributo de longitud de la variable del lenguaje principal y un puntero para el almacenamiento intermedio que contiene la longitud real.<sup>114</sup> Si se utilizan localizadores o variables de referencia a archivos para representar los LOB, estas entradas no son necesarias.

En las SQLDA que contienen ambos tipos de entradas, las SQLVAR base están en un bloque antes del bloque de SQLVAR secundarias. En cada una, el número de entradas es igual al valor de SQLD (incluso aunque muchas de las entradas SQLVAR secundarias pueden estar sin utilizar).

Las circunstancias bajo las que DESCRIBE establece las entradas SQLVAR se detallan en el apartado “Efecto de DESCRIBE en la SQLDA” en la página 1274.

---

114. La información de tipo diferenciado y de LOB no se solapa, por lo tanto los tipos diferenciados pueden estar basados en LOB sin hacer que se triplique el número de entradas de SQLVAR en DESCRIBE.

## Campos en la cabecera SQLDA

Tabla 38. Campos en la cabecera SQLDA

| Nombre C | Tipo de datos SQL | Uso en DESCRIBE y PREPARE (establecido por el gestor de bases de datos excepto para SQLN)                                                                                                                                                                                                                                                                                                                              | Uso en FETCH, OPEN, EXECUTE y CALL (establecido por la aplicación antes de ejecutar la sentencia)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|----------|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| sqldaid  | CHAR(8)           | El séptimo byte de este campo es un byte de distintivo llamado SQLDOUBLED. El gestor de bases de datos establece SQLDOUBLED en el carácter '2' si se han creado dos entradas SQLVAR para cada columna; de lo contrario, se establecen en un blanco (X'20' en ASCII, X'40' en EBCDIC). Consulte el apartado "Efecto de DESCRIBE en la SQLDA" en la página 1274 para ver los detalles de cuándo se establece SQLDOUBLED. | El séptimo byte de este campo se utiliza cuando se dobla el número de SQLVAR. Se denomina SQLDOUBLED. Si alguna de las variables del lenguaje principal que se describen es un tipo estructurado, BLOB, CLOB o DBCLOB, el séptimo byte debe establecerse en el carácter '2'; en otro caso puede establecerse en cualquier carácter, pero es aconsejable utilizar un espacio en blanco.<br><br>Cuando se utiliza con la sentencia CALL y una o varias SQLVAR definen el campo de datos como FOR BIT DATA, el sexto byte debe establecerse en el carácter '+'; en otro caso puede establecerse en cualquier carácter, pero es aconsejable utilizar un espacio en blanco. |
| sqldabc  | INTEGER           | Para 32 bits, la longitud de la SQLDA, que es igual a SQLN*44+16. Para 64 bits, la longitud de la SQLDA, que es igual a SQLN*56+16                                                                                                                                                                                                                                                                                     | Para 32 bits, la longitud de la SQLDA, que es $\geq$ SQLN*44+16. Para 64 bits, la longitud de la SQLDA, que es $\geq$ SQLN*56+16.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| sqln     | SMALLINT          | Sin cambiar por el gestor de bases de datos. Debe estar establecido en un valor mayor o igual que cero antes de que se ejecute la sentencia DESCRIBE. Indica el número total de ocurrencias de SQLVAR.                                                                                                                                                                                                                 | Número total de ocurrencias de SQLVAR proporcionadas en la SQLDA. SQLN debe estar establecido en un valor mayor o igual que cero.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| sqld     | SMALLINT          | Establecido por el gestor de bases de datos en el número de columnas de la tabla resultante (o en cero si la sentencia que se describe no es una sentencia-select).                                                                                                                                                                                                                                                    | El número de variables del lenguaje principal descritas por las ocurrencias de SQLVAR.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

## Campos de una ocurrencia de una SQLVAR base

Tabla 39. Campos en una SQLVAR base

| Nombre  | Tipo de datos | Uso en DESCRIBE y PREPARE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Uso en FETCH, OPEN, EXECUTE y CALL                                                                                                                                                                                                                                                                                                             |
|---------|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| sqltype | SMALLINT      | <p>Indica el tipo de datos de la columna y si puede contener nulos. Tabla 41 en la página 1276 lista todos los valores permitidos y sus significados.</p> <p>Observe que para un tipo de referencia diferenciado, el tipo de datos del tipo base se coloca en este campo. Para un tipo estructurado, el tipo de datos del resultado de la función de transformación FROM SQL del grupo de transformación (basado en el registro especial CURRENT DEFAULT TRANSFORM GROUP) se coloca en este campo. No existe ninguna indicación en la SQLVAR base de que forma parte de la descripción de un tipo definido por el usuario o tipo de referencia.</p> | <p>Igual que para la variable del lenguaje principal. Las variables del lenguaje principal para los valores de fecha y hora deben ser variables de serie de caracteres. Para FETCH, un código de tipo de fecha y hora significa una serie de caracteres de longitud fija. Si sqltype es un valor de número par, se ignora el campo sqlind.</p> |
| sqllen  | SMALLINT      | <p>El atributo de longitud de la columna. Para columnas de fecha y hora, la longitud de la representación de serie de los valores. Vea Tabla 41 en la página 1276.</p> <p>Observe que el valor está establecido en 0 para las series de gran objeto (incluso para aquellas cuyo atributo de longitud sea lo suficientemente pequeño como para caber en un entero de dos bytes).</p>                                                                                                                                                                                                                                                                 | <p>El atributo de longitud de la variable del lenguaje principal. Vea Tabla 41 en la página 1276.</p> <p>Observe que el gestor de bases de datos ignora el valor para las columnas CLOB, DBCLOB y BLOB. Se utiliza el campo len.sqllonglen de la SQLVAR secundaria en su lugar.</p>                                                            |

Tabla 39. Campos en una SQLVAR base (continuación)

| Nombre  | Tipo de datos | Uso en DESCRIBE y PREPARE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Uso en FETCH, OPEN, EXECUTE y CALL                                                                                                                                               |
|---------|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| sqldata | puntero       | <p>En las SQLVAR de serie-caracteres, sqldata contiene 0 si la columna está definida con el atributo FOR BIT DATA. Si la columna no tiene el atributo FOR BIT DATA, el valor depende de la codificación de los datos. Para datos codificados SBCS de un solo byte, sqldata contiene la página de códigos SBCS. Para datos codificados DBCS mixtos, sqldata contiene la página de códigos SBCS asociada con la página de códigos DBCS compuesta. Para los datos codificados de EUC japonés o chino tradicional, sqldata contiene la página de códigos EUC compuesta.</p> <p>Para todos los tipos de columna, sqldata es indefinido.</p> | <p>Contiene la dirección de la variable del lenguaje principal (donde se almacenarán los datos leídos).</p>                                                                      |
| sqlind  | puntero       | <p>En las SQLVAR de serie-caracteres, sqlind contiene 0 excepto para los datos codificados DBCS mixtos en que sqlind contiene la página de códigos DBCS asociada con la página de códigos DBCS compuesta.</p> <p>Para el resto de tipos de columna, sqlind no está definido.</p>                                                                                                                                                                                                                                                                                                                                                       | <p>Contiene la dirección de una variable indicadora asociada si existe una; de lo contrario, no se utiliza. Si sqltype es un valor de número par, se ignora el campo sqlind.</p> |

## SQLDA

Tabla 39. Campos en una SQLVAR base (continuación)

| Nombre  | Tipo de datos | Uso en DESCRIBE y PREPARE                                                                                                                                                                                                                                                                                                                                                                          | Uso en FETCH, OPEN, EXECUTE y CALL                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---------|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| sqlname | VARCHAR (30)  | <p>Contiene el nombre no calificado de la columna.</p> <p>Para las columnas que tienen un nombre generado por el sistema (la columna del resultado no se ha derivado directamente de una sola columna y no ha especificado ningún nombre que utilice la cláusula AS), el trigésimo byte se establece en X'FF'. Para los nombres de columna especificados por la cláusula AS, el byte es X'00'.</p> | <p>Cuando se utiliza con la sentencia CALL para acceder al servidor de aplicaciones DRDA, sqlname se puede establecer para indicar una serie FOR BIT DATA tal como sigue:</p> <ul style="list-style-type: none"> <li>• la longitud de sqlname es 8</li> <li>• los cuatro primeros bytes de sqlname son X'00000000'</li> <li>• los cuatro bytes restantes de sqlname están reservados (y se ignoran actualmente).</li> </ul> <p>Además, sqltype debe indicar un CHAR, VARCHAR o LONG VARCHAR y el sexto byte del campo sqldaid se establece en el carácter '+'.</p> <p>Esta técnica también puede utilizarse con OPEN y EXECUTE cuando se utiliza DB2 Connect para acceder al servidor.</p> |

## Campos de una ocurrencia de una SQLVAR secundaria

Tabla 40. Campos en una SQLVAR secundaria

| Nombre         | Tipo de datos                                  | Uso en DESCRIBE y PREPARE                                   | Uso en FETCH, OPEN, EXECUTE y CALL                                                                                                                                                                                                                                                                   |
|----------------|------------------------------------------------|-------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| len.sqllonglen | INTEGER                                        | El atributo de longitud de una columna BLOB, CLOB o DBCLOB. | El atributo de longitud de una variable del lenguaje principal BLOB, CLOB o DBCLOB. El gestor de bases de datos pasa por alto el campo SQLLEN de la SQLVAR base para los tipos de datos. El atributo de longitud almacena el número de bytes para BLOB o CLOB y el número de caracteres para DBCLOB. |
| reservado2     | CHAR(3) para 32 bits, y CHAR(11) para 64 bits. | No utilizado.                                               | No utilizado.                                                                                                                                                                                                                                                                                        |

Tabla 40. Campos en una SQLVAR secundaria (continuación)

| Nombre      | Tipo de datos | Uso en DESCRIBE y PREPARE                                                                                                                                                                                                                                                                    | Uso en FETCH, OPEN, EXECUTE y CALL                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-------------|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| sqlflag4    | CHAR(1)       | El valor es X'01' si la SQLVAR representa un tipo de destino con un tipo de destino denominado en sqldatatype_name. El valor es X'12' si SQLVAR representa un tipo estructurado, y nombre_tipoDatosSql contiene el nombre del tipo definido por el usuario. En otro caso, el valor es X'00'. | Se establece en X'01' si la SQLVAR representa un tipo de destino mencionado en nombre_tipoDatosSql. El valor es X'12' si SQLVAR representa un tipo estructurado, y nombre_tipoDatosSql contiene el nombre del tipo definido por el usuario. En otro caso, el valor es X'00'.                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| sqldataalen | puntero       | No utilizado.                                                                                                                                                                                                                                                                                | <p>Utilizado solamente para las variables del lenguaje principal BLOB, CLOB y DBCLOB.</p> <p>Si este campo es NULL, entonces la longitud real (en caracteres) debe almacenarse en los 4 bytes inmediatamente antes del inicio de los datos y SQLDATA debe apuntar al primer byte de la longitud de campo.</p> <p>Si este campo no es NULL, contiene un puntero para un almacenamiento intermedio de 4 bytes de longitud que contiene la longitud real <i>en bytes</i> (incluso para DBCLOB) de los datos del almacenamiento intermedio al que apunta el campo de SQLDATA de la SQLVAR base coincidente.</p> <p>Observe que, sin tener en cuenta si este campo se utiliza o no, debe establecerse el campo len.sqllonglen.</p> |

## SQLDA

Tabla 40. Campos en una SQLVAR secundaria (continuación)

| Nombre             | Tipo de datos | Uso en DESCRIBE y PREPARE                                                                                                                                                                                                                                                                                                                              | Uso en FETCH, OPEN, EXECUTE y CALL                                                                                                                                                        |
|--------------------|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| nombre_sqldatatype | VARCHAR(27)   | Para una columna de tipo definido por el usuario, el gestor de bases de datos establece este campo en el nombre del tipo definido por el usuario, calificado al completo. <sup>1</sup> Para un tipo de referencia, el gestor de bases de datos establece este campo en el nombre del tipo calificado al completo del tipo de destino de la referencia. | Para los tipos estructurados, este campo se establece en el nombre del tipo definido por el usuario, calificado al completo, con el formato indicado en la nota de la tabla. <sup>1</sup> |
| reservado          | CHAR(3)       | No utilizado.                                                                                                                                                                                                                                                                                                                                          | No utilizado.                                                                                                                                                                             |

### Nota:

1. Los 8 primeros bytes contienen el nombre de esquema del tipo (ampliado por la derecha con espacios, si es necesario). El byte 9 contiene un punto (.). Los bytes del 10 al 27 contienen la parte de orden inferior del nombre de tipo que *no* se extiende por la derecha con espacios.

Tenga en cuenta que, aunque el principal propósito de este campo es para el nombre de los tipos definidos por el usuario, el campo también se establece para los tipos de datos predefinidos por IBM. En este caso, el nombre de esquema es SYSIBM y la parte de orden inferior del nombre es el nombre almacenado en la columna TYPENAME de la vista de catálogo DATATYPES. Por ejemplo:

| Nombre tipo     | longitud | nombre_sqldatatype |
|-----------------|----------|--------------------|
| -----           | -----    | -----              |
| A.B             | 10       | A .B               |
| INTEGER         | 16       | SYSIBM .INTEGER    |
| "Frank's".SMINT | 13       | Frank's .SMINT     |
| MY."type "      | 15       | MY .type           |

## Efecto de DESCRIBE en la SQLDA

Para una sentencia DESCRIBE o PREPARE INTO, el gestor de base de datos siempre establece SQLD en el número de columnas del conjunto resultante.

Las SQLVAR de la SQLDA se establecen en los casos siguientes:

- SQLN >= SQLD y ninguna columna es un LOB, un tipo definido por el usuario ni un tipo de referencia

Se establecen las primeras entradas SQLD SQLVAR y SQLDOUBLED se establece en blanco.



- $SQLN \geq 2 * SQLD$  y como mínimo una columna es un LOB, un tipo definido por el usuario o un tipo de referencia

Las entradas SQLD SQLVAR se establecen dos veces y SQLDOUBLED se establece en '2'.

- $SQLD \leq SQLN < 2 * SQLD$  y como mínimo una columna es un tipo diferenciado o un tipo de referencia, pero no hay columnas LOB ni columnas de tipo estructurado

Se establecen las primeras entradas SQLD SQLVAR y SQLDOUBLED se establece en blanco. Si la opción SQLWARN es YES, se emite un aviso SQLCODE +237 (SQLSTATE 01594).

Las SQLVAR de la SQLDA NO se establecen (es necesaria la asignación de espacio adicional y otra DESCRIBE) en los casos siguientes:

- $SQLN < SQLD$  y ninguna columna es un LOB, un tipo definido por el usuario ni un tipo de referencia

No se establece ninguna entrada SQLVAR y SQLDOUBLED se establece en blanco. Si la opción SQLWARN es YES, se emite un aviso SQLCODE +236 (SQLSTATE 01005).

Asigne las SQLD SQLVAR para una operación DESCRIBE satisfactoria.

- $SQLN < SQLD$  y como mínimo una columna es un tipo diferenciado o un tipo de referencia, pero no hay columnas LOB ni columnas de tipo estructurado

No se establece ninguna entrada SQLVAR y SQLDOUBLED se establece en blanco. Si la opción SQLWARN es YES, se emite un aviso SQLCODE +239 (SQLSTATE 01005).

Asigne un número de estructuras SQLVAR igual a  $2 * SQLD$  para lograr una operación DESCRIBE satisfactoria que incluya los nombres de los tipos diferenciados y tipos destino de tipos de referencia.

- $SQLN < 2 * SQLD$  y como mínimo una columna es un LOB o un tipo estructurado

No se establece ninguna entrada SQLVAR y SQLDOUBLED se establece en blanco. Se emite un aviso SQLCODE +238 (SQLSTATE 01005) (sin tener en cuenta el valor de la opción de enlace lógico SQLWARN).

Asigne las  $2 * SQLD$  SQLVAR para una operación DESCRIBE satisfactoria.

Las referencias de las listas anteriores a columnas LOB incluyen las columnas de tipo diferenciado cuyo tipo fuente es un tipo LOB.

La opción SQLWARN del mandato BIND o PREP se utiliza para controlar si DESCRIBE (o PREPARE INTO) devolverá los SQLCODE de aviso +236, +237, +239. Se recomienda que el código de la aplicación tenga siempre en cuenta que podrían devolverse estos SQLCODE. El SQLCODE de aviso +238 siempre se devuelve cuando hay columnas LOB o de tipo estructurado en la lista de

selección y no hay suficientes estructuras SQLVAR en la SQLDA. Es la única manera de que la aplicación pueda saber el número de estructuras SQLVAR que deben doblarse debido a que hay una columna LOB o de tipo estructurado en el conjunto resultante.

Si se está describiendo un tipo estructurado, pero no se define ninguna transformación FROM SQL (porque no se especificó ningún TRANSFORM GROUP utilizando el registro especial CURRENT DEFAULT TRANSFORM GROUP (SQLSTATE 42741), o porque el grupo mencionado no tiene una función de transformación FROM SQL definida (SQLSTATE 42744), DESCRIBE emitirá un error. Este error es el mismo que se devuelve para un DESCRIBE de una tabla que tenga una columna de tipo estructurado.

## SQLTYPE y SQLLEN

La Tabla 41 muestra los valores que pueden aparecer en los campos SQLTYPE y SQLLEN de la SQLDA. En DESCRIBE y PREPARE INTO, un valor par de SQLTYPE significa que la columna no permite nulos y un valor impar significa que la columna permite nulos. En FETCH, OPEN, EXECUTE y CALL, un valor par de SQLTYPE significa que no se proporciona una variable indicadora y un valor impar significa que SQLLIND contiene la dirección de una variable indicadora.

Tabla 41. Valores SQLTYPE y SQLLEN para DESCRIBE, FETCH, OPEN, EXECUTE y CALL

| SQLTYPE | Para DESCRIBE y PREPARE INTO |                                    | Para FETCH, OPEN, EXECUTE y CALL                                                    |                                                            |
|---------|------------------------------|------------------------------------|-------------------------------------------------------------------------------------|------------------------------------------------------------|
|         | Tipo de datos de columna     | SQLLEN                             | Tipo de datos de variable del lenguaje principal                                    | SQLLEN                                                     |
| 384/385 | date                         | 10                                 | representación de una fecha en serie de caracteres de longitud fija                 | atributo de longitud de la variable del lenguaje principal |
| 388/389 | time                         | 8                                  | representación de una hora en serie de caracteres de longitud fija                  | atributo de longitud de la variable del lenguaje principal |
| 392/393 | timestamp                    | 26                                 | representación de una indicación de la hora en serie de caracteres de longitud fija | atributo de longitud de la variable del lenguaje principal |
| 396/397 | DATALINK                     | atributo de longitud de la columna | DATALINK                                                                            | atributo de longitud de la variable del lenguaje principal |

Tabla 41. Valores *SQLTYPE* y *SQLLEN* para *DESCRIBE*, *FETCH*, *OPEN*, *EXECUTE* y *CALL* (continuación)

| SQLTYPE | Para <i>DESCRIBE</i> y <i>PREPARE INTO</i>     |                                    | Para <i>FETCH</i> , <i>OPEN</i> , <i>EXECUTE</i> y <i>CALL</i> |                                                            |
|---------|------------------------------------------------|------------------------------------|----------------------------------------------------------------|------------------------------------------------------------|
|         | Tipo de datos de columna                       | SQLLEN                             | Tipo de datos de variable del lenguaje principal               | SQLLEN                                                     |
| 400/401 | N/D                                            | N/D                                | serie gráfica terminada en NUL                                 | atributo de longitud de la variable del lenguaje principal |
| 404/405 | BLOB                                           | 0 *                                | BLOB                                                           | No utilizado. *                                            |
| 408/409 | CLOB                                           | 0 *                                | CLOB                                                           | No utilizado. *                                            |
| 412/413 | DBCLOB                                         | 0 *                                | DBCLOB                                                         | No utilizado. *                                            |
| 448/449 | serie de caracteres de longitud variable       | atributo de longitud de la columna | serie de caracteres de longitud variable                       | atributo de longitud de la variable del lenguaje principal |
| 452/453 | serie de caracteres de longitud fija           | atributo de longitud de la columna | serie de caracteres de longitud fija                           | atributo de longitud de la variable del lenguaje principal |
| 456/457 | serie larga de caracteres de longitud variable | atributo de longitud de la columna | serie larga de caracteres de longitud variable                 | atributo de longitud de la variable del lenguaje principal |
| 460/461 | N/D                                            | N/D                                | serie de caracteres terminada en NUL                           | atributo de longitud de la variable del lenguaje principal |
| 464/465 | serie gráfica de longitud variable             | atributo de longitud de la columna | serie gráfica de longitud variable                             | atributo de longitud de la variable del lenguaje principal |
| 468/469 | serie gráfica de longitud fija                 | atributo de longitud de la columna | serie gráfica de longitud fija                                 | atributo de longitud de la variable del lenguaje principal |
| 472/473 | serie gráfica de longitud variable larga       | atributo de longitud de la columna | serie gráfica larga                                            | atributo de longitud de la variable del lenguaje principal |

## SQLDA

Tabla 41. Valores SQLTYPE y SQLLEN para DESCRIBE, FETCH, OPEN, EXECUTE y CALL (continuación)

| SQLTYPE | Para DESCRIBE y PREPARE INTO |                                                 | Para FETCH, OPEN, EXECUTE y CALL                 |                                                 |
|---------|------------------------------|-------------------------------------------------|--------------------------------------------------|-------------------------------------------------|
|         | Tipo de datos de columna     | SQLLEN                                          | Tipo de datos de variable del lenguaje principal | SQLLEN                                          |
| 480/481 | coma flotante                | 8 para precisión doble, 4 para precisión simple | coma flotante                                    | 8 para precisión doble, 4 para precisión simple |
| 484/485 | decimal empaquetado          | precisión en byte 1; escala en byte 2           | decimal empaquetado                              | precisión en byte 1; escala en byte 2           |
| 492/493 | entero superior              | 8                                               | entero superior                                  | 8                                               |
| 496/497 | entero grande                | 4                                               | entero grande                                    | 4                                               |
| 500/501 | entero pequeño               | 2                                               | entero pequeño                                   | 2                                               |
| 916/917 | No aplicable                 | No aplicable                                    | variable de referencia a archivos BLOB.          | 267                                             |
| 920/921 | No aplicable                 | No aplicable                                    | variable de referencia a archivos CLOB.          | 267                                             |
| 924/925 | No aplicable                 | No aplicable                                    | variable de referencia a archivos DBCLOB.        | 267                                             |
| 960/961 | No aplicable                 | No aplicable                                    | localizador de BLOB                              | 4                                               |
| 964/965 | No aplicable                 | No aplicable                                    | localizador de CLOB                              | 4                                               |
| 968/969 | No aplicable                 | No aplicable                                    | localizador de DBCLOB                            | 4                                               |

### Nota:

- El campo len.sqlllonglen de la SQLVAR secundaria contiene el atributo de longitud de la columna.
- SQLTYPE se ha modificado desde la versión anterior para portabilidad en DB2. Los valores de las versiones anteriores (consulte Referencia SQL de la versión anterior) seguirán siendo soportados.

### SQLTYPES no soportados y no reconocibles

Los valores que aparecen en el campo SQLTYPE de SQLDA dependen del nivel de soporte de tipo de datos disponible en el encargado de enviar los datos así como en el que los recibe. Esto es especialmente importante cuando se añaden nuevos tipos de datos al producto.

Los tipos de datos nuevos pueden recibir o no soporte del que envía los datos o del que los recibe y pueden estar reconocidos o incluso no estarlo por el que envía los datos o el que los recibe. Según la situación, puede devolverse el tipo de datos nuevo o puede devolverse un tipo de datos compatible con el acuerdo del que envía los datos y del que los recibe o bien puede dar como resultado un error.

Cuando el que envía los datos y el que los recibe se ponen de acuerdo para utilizar un tipo de datos compatible, la indicación siguiente expresa la correlación que tendrá lugar. Esta correlación tendrá lugar cuando, como mínimo, o el que envía los datos o el que los recibe no dé soporte al tipo de datos proporcionado. Tanto la aplicación como el gestor de bases de datos pueden proporcionar el tipo de datos no soportado.

| Tipo de datos | Tipo de datos compatible                |
|---------------|-----------------------------------------|
| BIGINT        | DECIMAL(19, 0)                          |
| ROWID         | VARCHAR(40) FOR BIT DATA <sup>115</sup> |

Tenga en cuenta que en SQLDA no se proporciona ninguna indicación de que se sustituya el tipo de datos.

### Números decimales empaquetados

Los números decimales empaquetados se almacenan en una variación de la notación Decimal codificado en binario (BCD). En BCD, cada nybble (cuatro bits) representa un dígito decimal. Por ejemplo, 0001 0111 1001 representa 179. Por lo tanto, se lee un valor decimal empaquetado nybble a nybble. Se almacena el valor en bytes y después se lee estos bytes en representación hexadecimal para volver a decimal. Por ejemplo, 0001 0111 1001 se convierte en 00000001 01111001 en la representación binaria. Leyendo este número como hexadecimal, se convierte en 0179.

La coma decimal se determina por la escala. En el caso de una columna DEC(12,5), por ejemplo, los 5 dígitos más a la derecha están a la derecha de la coma decimal.

El signo lo indica un nybble a la derecha de los nybbles que representa los dígitos. Un signo positivo o negativo se indica de la siguiente manera:

115. ROWID está soportado por DB2 Universal Database para OS/390 Versión 6.

Tabla 42. Valores para el indicador de signo de un número decimal empaquetado

| Signo        | Representación |         |             |
|--------------|----------------|---------|-------------|
|              | Binaria        | Decimal | Hexadecimal |
| Positivo (+) | 1100           | 12      | C           |
| Negativo (-) | 1101           | 13      | D           |

En resumen:

1. Para almacenar cualquier valor, asigne  $p/2+1$  bytes, donde  $p$  es la precisión.
2. Asigne los nybbles de izquierda a derecha para representar el valor. Si un número tiene una precisión par, se añade un nybble de cero inicial. Esta asignación incluye los dígitos cero iniciales (sin significado) y de cola (significativos).
3. El nybble de signo será el segundo nybble del último byte.

Existe una manera alternativa de realizar conversiones de decimales empaquetados, consulte el apartado "CHAR" en la página 289.

Por ejemplo:

| Columna  | Valor   | Nybbles en hexadecimal agrupados por bytes |
|----------|---------|--------------------------------------------|
| DEC(8,3) | 6574,23 | 00 65 74 23 0C                             |
| DEC(6,2) | -334,02 | 00 33 40 2D                                |
| DEC(7,5) | 5,2323  | 05 23 23 0C                                |
| DEC(5,2) | -23,5   | 02 35 0D                                   |

### Campo SQLLEN para decimal

El campo SQLLEN contiene la precisión (primer byte) y la escala (segundo byte) de la columna decimal. Al escribir una aplicación portable, los bytes de la precisión y de la escala se deben establecer individualmente, en lugar de establecerlos conjuntamente como un entero corto. Esto evitará los problemas en la inversión de bytes de enteros.

Por ejemplo, en C:

```
((char *)&(sqlda->sqlvar[i].sqllen))[0] = precision;
((char *)&(sqlda->sqlvar[i].sqllen))[1] = scale;
```

---

## Apéndice D. Vistas de catálogo

El gestor de bases de datos crea y mantiene dos conjuntos de vistas de catálogo del sistema. Este apéndice contiene una descripción de cada vista de catálogo del sistema, incluyendo los nombres de columna y los tipos de datos. Todas las vistas de catálogo del sistema se crean cuando se crea la base de datos con el mandato CREATE DATABASE. Las vistas de catálogo no pueden crearse ni eliminarse explícitamente. Las vistas de catálogo del sistema se actualizan durante el funcionamiento normal en respuesta a las sentencias de definición de datos SQL, rutinas de entorno y algunos programas de utilidad. Los datos de las vistas de catálogo del sistema están disponibles mediante las funciones de la consulta SQL normal. Las vistas de catálogo del sistema no se pueden modificar utilizando mandatos de manipulación de datos SQL normales, a excepción de algunas vistas específicas del catálogo que se puedan actualizar.

Se da soporte a las vistas de catálogo y a las tablas base de catálogo en la Versión 1. Las vistas están dentro del esquema SYSCAT y por omisión se otorga a PUBLIC el privilegio SELECT para todas las vistas. Los programas de aplicación deben escribirse en estas vistas en lugar de en las tablas base del catálogo. Un segundo conjunto de vistas formadas a partir de un subconjunto de las que están en el esquema SYSCAT contienen información de estadísticas utilizadas por el optimizador. Las vistas del esquema SYSSTAT contienen algunas columnas actualizables.

**Aviso:** La intención es habilitar aplicaciones para actualizar algunas columnas utilizando las vistas SYSSTAT, pero teniendo las vistas SYSCAT de sólo lectura. Actualmente, las vistas SYSCAT no son sólo de lectura. Se avisa a los desarrolladores de aplicaciones de que se aseguren de que las aplicaciones se graben sólo para actualizar información de catálogo utilizando las vistas SYSSTAT. Las vistas SYSCAT pasarán a ser vistas de sólo lectura después de la migración de la versión siguiente.

Las vistas de catálogo están diseñadas para utilizar convenios más coherentes que las tablas base del catálogo principales. Como tales, el orden de las columnas puede cambiar de un release a otro. Para evitar que esto afecta a la lógica de programación, especifique siempre explícitamente las columnas en una lista de selección en lugar de utilizar el valor por omisión mediante SELECT \*. Las columnas tienen nombres uniformes basados en el tipo de objetos que describen:

**Objeto descrito**

**Nombres de columna**

## Vistas de catálogo

|                                           |                              |
|-------------------------------------------|------------------------------|
| Tabla                                     | TABSCHEMA, TABNAME           |
| Índice                                    | INDSCHEMA, INDNAME           |
| Vista                                     | VIEWSCHEMA, VIEWNAME         |
| Restricción                               | CONSTSCHEMA, CONSTNAME       |
| Desencadenante                            | TRIGSCHEMA, TRIGNAME         |
| Paquete                                   | PKGSCHEMA, PKGNAME           |
| Tipo                                      | TYPESCHEMA, TYPENAME, TYPEID |
| Función                                   | FUNCSCHEMA, FUNCNAME, FUNCID |
| Columna                                   | COLNAME                      |
| Esquema                                   | SCHEMANAME                   |
| Espacio de tablas                         | TBSPACE                      |
| Grupo de nodos                            | NGNAME                       |
| Agrupación de almacenamientos intermedios | BPNAME                       |
| Supervisor de sucesos                     | EVMONNAME                    |
| Indicación de la hora de creación         | CREATE_TIME                  |

- “Vistas de catálogo actualizables”
- “‘Guía’ de las vistas de catálogo”
- “‘Guía’ de las vistas de catálogo actualizables” en la página 1285

---

## Vistas de catálogo actualizables

Las vistas actualizables contienen información de estadísticas utilizada por el optimizador. Algunas columnas de estas vistas pueden cambiarse para investigar el rendimiento de bases de datos hipotéticas. Un objeto (tabla, columna, función o índice) sólo aparecerá en la vista de catálogo actualizable para un usuario determinado si dicho usuario ha creado el objeto, tiene el privilegio CONTROL en el objeto o tiene el privilegio DBADM explícito. Estas vistas se encuentran en el esquema SYSSTAT. Se definen al principio de las tablas de base de catálogo del sistema.

Antes de cambiar cualquier estadística por primera vez, es aconsejable emitir el mandato RUNSTATS para que todas las estadísticas reflejen el estado actual.

---

## ‘Guía’ de las vistas de catálogo

| Descripción                               | Vista de catálogo | Página |
|-------------------------------------------|-------------------|--------|
| atributos de tipos de datos estructurados | SYSCAT.ATTRIBUTES | 1287   |
| autorizaciones en base de datos           | SYSCAT.DBAUTH     | 1306   |



| Descripción                                                                | Vista de catálogo         | Página |
|----------------------------------------------------------------------------|---------------------------|--------|
| configuración de agrupación de almacenamiento intermedio en grupo de nodos | SYSCAT.BUFFERPOOLS        | 1290   |
| tamaño de agrupaciones de almacenamiento intermedio en nodo                | SYSCAT.BUFFERPOOLNODES    | 1289   |
| funciones de conversión                                                    | SYSCAT.CASTFUNCTIONS      | 1291   |
| restricciones de comprobación                                              | SYSCAT.CHECKS             | 1292   |
| privilegios de columna                                                     | SYSCAT.COLAUTH            | 1293   |
| columnas                                                                   | SYSCAT.COLUMNS            | 1297   |
| columnas referenciadas por restricciones de comprobación                   | SYSCAT.COLCHECKS          | 1294   |
| columnas utilizadas en claves                                              | SYSCAT.KEYCOLUSE          | 1333   |
| opciones de columna detalladas                                             | SYSCAT.COLOPTIONS         | 1296   |
| estadísticas detalladas de columna                                         | SYSCAT.COLDIST            | 1295   |
| dependencias de restricción                                                | SYSCAT.CONSTDEP           | 1303   |
| tipos de datos                                                             | SYSCAT.DATATYPES          | 1304   |
| definiciones de supervisor de sucesos                                      | SYSCAT.EVENTMONITORS      | 1308   |
| sucesos supervisados actualmente                                           | SYSCAT.EVENTS             | 1310   |
| jerarquías (tipos, tablas, vistas)                                         | SYSCAT.FULLHIERARCHIES    | 1311   |
| dependencias de funciones                                                  | SYSCAT.FUNCDEP            | 1312   |
| correlación de funciones                                                   | SYSCAT.FUNCMAPPINGS       | 1315   |
| opciones de correlación de funciones                                       | SYSCAT.FUNCMAPOPTIONS     | 1313   |
| opciones de parámetro de correlación de funciones                          | SYSCAT.FUNCMAPPARMOPTIONS | 1314   |
| parámetros de función                                                      | SYSCAT.FUNCPARMS          | 1316   |
| jerarquías (tipos, tablas, vistas)                                         | SYSCAT.HIERARCHIES        | 1324   |
| privilegios de índice                                                      | SYSCAT.INDEXAUTH          | 1325   |
| Columnas de índice                                                         | SYSCAT.INDEXCOLUSE        | 1326   |
| dependencias de índice                                                     | SYSCAT.INDEXDEP           | 1327   |
| índices                                                                    | SYSCAT.INDEXES            | 1328   |
| definiciones de grupo de nodos                                             | SYSCAT.NODEGROUPS         | 1336   |
| nodos de grupo de nodos                                                    | SYSCAT.NODEGROUPDEF       | 1335   |
| correlación de objeto                                                      | SYSCAT.NAMEMAPPINGS       | 1334   |
| dependencias de paquete                                                    | SYSCAT.PACKAGEDEP         | 1338   |
| privilegios de paquete                                                     | SYSCAT.PACKAGEAUTH        | 1337   |

## Vistas de catálogo

| Descripción                                                   | Vista de catálogo      | Página |
|---------------------------------------------------------------|------------------------|--------|
| paquetes                                                      | SYSCAT.PACKAGES        | 1339   |
| correlaciones de particiones                                  | SYSCAT.PARTITIONMAPS   | 1344   |
| privilegios de paso a través                                  | SYSCAT.PASSTHROUGH     | 1345   |
| opciones de procedimiento                                     | SYSCAT.PROCOPTIONS     | 1349   |
| opciones de parámetro de procedimiento                        | SYSCAT.PROCPARMOPTIONS | 1350   |
| parámetros de procedimiento                                   | SYSCAT.PROCPARMS       | 1351   |
| proporciona compatibilidad DB2 Universal Database para OS/390 | SYSIBM.SYSDUMMY1       | 1286   |
| restricciones de referencia                                   | SYSCAT.REFERENCES      | 1353   |
| opciones de tabla remota                                      | SYSCAT.TABOPTIONS      | 1373   |
| correlación de tipos de datos invertida                       | SYSCAT.REVTYPEMAPPINGS | 1354   |
| privilegios de esquema                                        | SYSCAT.SCHEMAAUTH      | 1356   |
| esquemas                                                      | SYSCAT.SCHEMATA        | 1357   |
| secuencias                                                    | SYSCAT.SEQUENCES       | 1358   |
| opciones de servidor                                          | SYSCAT.SERVEROPTIONS   | 1360   |
| valores de opciones de servidor                               | SYSCAT.USEROPTIONS     | 1380   |
| sentencias en paquetes                                        | SYSCAT.STATEMENTS      | 1362   |
| procedimientos almacenados                                    | SYSCAT.PROCEDURES      | 1346   |
| servidores del sistema                                        | SYSCAT.SERVERS         | 1361   |
| restricciones de tabla                                        | SYSCAT.TABCONST        | 1365   |
| privilegios de tabla                                          | SYSCAT.TABAUTH         | 1363   |
| tablas                                                        | SYSCAT.TABLES          | 1366   |
| espacios de tablas                                            | SYSCAT.TABLESPACES     | 1371   |
| privilegios de uso de espacios de tablas                      | SYSCAT.TBSPACEAUTH     | 1374   |
| dependencias de desencadenante                                | SYSCAT.TRIGDEP         | 1375   |
| desencadenantes                                               | SYSCAT.TRIGGERS        | 1376   |
| correlación de tipos                                          | SYSCAT.TYPEMAPPINGS    | 1378   |
| funciones definidas por el usuario                            | SYSCAT.FUNCTIONS       | 1318   |
| dependencias de vista                                         | SYSCAT.VIEWDEP         | 1381   |
| vistas                                                        | SYSCAT.TABLES          | 1366   |
|                                                               | SYSCAT.VIEWS           | 1382   |
| opciones de reiniciador                                       | SYSCAT.WRAPOPTIONS     | 1383   |

| <b>Descripción</b> | <b>Vista de catálogo</b> | <b>Página</b> |
|--------------------|--------------------------|---------------|
| reiniciadores      | SYSCAT.WRAPPERS          | 1384          |

---

**'Guía' de las vistas de catálogo actualizables**

| <b>Descripción</b>                 | <b>Vista de catálogo</b> | <b>Página</b> |
|------------------------------------|--------------------------|---------------|
| columnas                           | SYSSTAT.COLUMNS          | 1386          |
| índices                            | SYSSTAT.INDEXES          | 1390          |
| estadísticas detalladas de columna | SYSSTAT.COLDIST          | 1385          |
| tablas                             | SYSSTAT.TABLES           | 1394          |
| funciones definidas por el usuario | SYSSTAT.FUNCTIONS        | 1388          |

## SYSIBM.SYSDUMMY1

---

### SYSIBM.SYSDUMMY1

Contiene una fila. Esta vista está disponible para las aplicaciones que necesitan compatibilidad con DB2 Universal Database para OS/390.

*Tabla 43. Vista de catálogo SYSCAT.DUMMY1*

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción |
|-------------------|---------------|----------------------|-------------|
| IBMREQD           | CHAR(1)       |                      | Y           |

## SYSCAT.ATTRIBUTES

Contiene una fila para cada atributo (incluidos los atributos heredados donde sea aplicable) que se define para un tipo de datos estructurado definido por el usuario.

Tabla 44. Vista de catálogo SYSCAT.ATTRIBUTES

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                                                                                                     |
|-------------------|---------------|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TYPESHEMA         | VARCHAR(128)  |                      | Nombre calificado del tipo de datos estructurado que incluye el atributo.                                                                                                                                                                                                                                                                       |
| TYPENAME          | VARCHAR(18)   |                      |                                                                                                                                                                                                                                                                                                                                                 |
| ATTR_NAME         | VARCHAR(18)   |                      | Nombre del atributo.                                                                                                                                                                                                                                                                                                                            |
| ATTR_TYPESHEMA    | VARCHAR(128)  |                      | Contiene el nombre calificado del tipo del atributo.                                                                                                                                                                                                                                                                                            |
| ATTR_TYPENAME     | VARCHAR(18)   |                      |                                                                                                                                                                                                                                                                                                                                                 |
| TARGET_TYPESHEMA  | VARCHAR(128)  |                      | Nombre calificado del tipo de destino si el tipo del atributo es REFERENCE. Valor nulo si el tipo del atributo no es REFERENCE.                                                                                                                                                                                                                 |
| TARGET_TYPENAME   | VARCHAR(18)   |                      |                                                                                                                                                                                                                                                                                                                                                 |
| SOURCE_TYPESHEMA  | VARCHAR(128)  |                      | Nombre calificado del tipo de datos en la jerarquía de tipos de datos en que se ha introducido el atributo. Para atributos no heredado, estas columnas son igual que TYPESHEMA y TYPENAME.                                                                                                                                                      |
| SOURCE_TYPENAME   | VARCHAR(18)   |                      |                                                                                                                                                                                                                                                                                                                                                 |
| ORDINAL           | SMALLINT      |                      | Posición del atributo en la definición del tipo de datos estructurado empezando por cero.                                                                                                                                                                                                                                                       |
| LENGTH            | INTEGER       |                      | Longitud máxima de datos. 0 para tipos diferenciados. La columna LENGTH indica la precisión para los campos DECIMAL.                                                                                                                                                                                                                            |
| SCALE             | SMALLINT      |                      | Escala para los campos DECIMAL; 0 si no es DECIMAL.                                                                                                                                                                                                                                                                                             |
| CODEPAGE          | SMALLINT      |                      | Página de códigos del atributo. Para los atributos de series de caracteres no definidos con FOR BIT DATA, el valor es la página de códigos de la base de datos. Para los atributos de series gráficas, el valor es la página de códigos DBCS implícita en la página de códigos de la base de datos (compuesta). De lo contrario, el valor es 0. |

## SYSCAT.ATTRIBUTES

Tabla 44. Vista de catálogo SYSCAT.ATTRIBUTES (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                                  |
|-------------------|---------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LOGGED            | CHAR(1)       |                      | Sólo se aplica a los atributos cuyo tipo sea LOB o un tipo diferenciado basado en LOB (de lo contrario, está en blanco).<br>Y = El atributo se anota cronológicamente.<br>N = El atributo no se anota cronológicamente.                                                      |
| COMPACT           | CHAR(1)       |                      | Sólo se aplica a los atributos cuyo tipo sea LOB o un tipo diferenciado basado en LOB (de lo contrario, está en blanco).<br>Y = El atributo se compacta en el almacenamiento.<br>N = El atributo no se compacta.                                                             |
| DL_FEATURES       | CHAR(10)      |                      | Sólo se aplica a los atributos de tipo DATALINK. Está en blanco para los atributos de tipo REFERENCE. En otro caso, es nulo. Codifica diversas características de DATALINK como, por ejemplo, tipo de enlace, modalidad de control, recuperación y propiedades de desenlace. |

---

**SYSCAT.BUFFERPOOLNODES**

Contiene una fila para cada nodo de la agrupación de almacenamientos intermedios para el que el tamaño de la agrupación de almacenamientos intermedios del nodo sea diferente del tamaño por omisión de la columna NPAGES de SYSCAT.BUFFERPOOLS.

*Tabla 45. Vista de catálogo SYSCAT.BUFFERPOOLNODES*

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                    |
|-------------------|---------------|----------------------|--------------------------------------------------------------------------------|
| BUFFERPOOLID      | INTEGER       |                      | Identificador interno de agrupación de almacenamientos intermedios             |
| NODENUM           | SMALLINT      |                      | Número de nodo                                                                 |
| NPAGES            | INTEGER       |                      | Número de páginas en la agrupación de almacenamientos intermedios en este nodo |

## SYSCAT.BUFFERPOOLS

---

### SYSCAT.BUFFERPOOLS

Contiene una fila para cada agrupación de almacenamientos intermedios de cada grupo de nodos.

Tabla 46. Vista de catálogo SYSCAT.BUFFERPOOLS

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                      |
|-------------------|---------------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BPNAME            | VARCHAR(18)   |                      | Nombre de la agrupación de almacenamientos intermedios                                                                                                                           |
| BUFFERPOOLID      | INTEGER       |                      | Identificador interno de agrupación de almacenamientos intermedios                                                                                                               |
| NGNAME            | VARCHAR(18)   | Sí                   | Nombre de grupo de nodos (NULL si la agrupación de almacenamientos intermedios existe en todos los nodos de la base de datos)                                                    |
| NPAGES            | INTEGER       |                      | Número de páginas de la agrupación de almacenamientos intermedios                                                                                                                |
| PAGESIZE          | INTEGER       |                      | Tamaño de página para esta agrupación de almacenamientos intermedios                                                                                                             |
| ESTORE            | CHAR(1)       |                      | N = Esta agrupación de almacenamientos intermedios no utiliza el almacenamiento ampliado.<br>Y = Esta agrupación de almacenamientos intermedios utiliza almacenamiento ampliado. |



**SYSCAT.CASTFUNCTIONS**

Contiene una fila para cada función de difusión. No incluye funciones de difusión incorporadas.

Tabla 47. Vista de catálogo SYSCAT.CASTFUNCTIONS

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                |
|-------------------|---------------|----------------------|----------------------------------------------------------------------------|
| FROM_TYPESHEMA    | VARCHAR(128)  |                      | Nombre calificado del tipo de datos del parámetro.                         |
| FROM_TYPENAME     | VARCHAR(18)   |                      |                                                                            |
| TO_TYPESHEMA      | VARCHAR(128)  |                      | Nombre calificado del tipo de datos del resultado después de la difusión.  |
| TO_TYPENAME       | VARCHAR(18)   |                      |                                                                            |
| FUNCSHEMA         | VARCHAR(128)  |                      | Nombre calificado de la función.                                           |
| FUNCNAME          | VARCHAR(18)   |                      |                                                                            |
| SPECIFICNAME      | VARCHAR(18)   |                      | El nombre de la instancia de función.                                      |
| ASSIGN_FUNCTION   | CHAR(1)       |                      | Y = Función de asignación implícita<br>N = No es una función de asignación |

## SYSCAT.CHECKS

---

### SYSCAT.CHECKS

Contiene una fila para cada restricción CHECK.

Tabla 48. Vista de catálogo SYSCAT.CHECKS

| Nombre de columna | Tipo de datos | Posibi-<br>lidad de<br>nulos | Descripción                                                                                                                                                                                                               |
|-------------------|---------------|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CONSTNAME         | VARCHAR(18)   |                              | Nombre de la restricción de comprobación (exclusivo en una tabla).                                                                                                                                                        |
| DEFINER           | VARCHAR(128)  |                              | ID de autorización bajo el cual se ha definido la restricción de comprobación.                                                                                                                                            |
| TABSCHEMA         | VARCHAR(128)  |                              | Nombre calificado de la tabla a la que se aplica esta restricción.                                                                                                                                                        |
| TABNAME           | VARCHAR(128)  |                              |                                                                                                                                                                                                                           |
| CREATE_TIME       | TIMESTAMP     |                              | La hora en la que se ha definido la restricción. Se utiliza para resolver las funciones que se utilizan en esta restricción. No se elegirá ninguna función que se haya creado después de la definición de la restricción. |
| QUALIFIER         | VARCHAR(128)  |                              | Valor del esquema por omisión en el momento de la definición de objeto. Se utiliza para completar cualquier referencia no calificada.                                                                                     |
| TYPE              | CHAR(1)       |                              | Tipo de restricción de comprobación:<br>A = El sistema generó una restricción de comprobación para una columna definida como GENERATED ALWAYS<br>C = Restricción de comprobación                                          |
| FUNC_PATH         | VARCHAR(254)  |                              | La vía de acceso de SQL que se ha utilizado cuando se ha creado la restricción.                                                                                                                                           |
| TEXT              | CLOB(64K)     |                              | El texto de la cláusula CHECK.                                                                                                                                                                                            |

## SYSCAT.COLAUTH

Contiene una o varias filas para cada usuario o grupo a los que se ha otorgado un privilegio a nivel de columna, que indican el tipo de privilegio y si se puede otorgar o no.

Tabla 49. Vista de catálogo SYSCAT.COLAUTH

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                    |
|-------------------|---------------|----------------------|--------------------------------------------------------------------------------------------------------------------------------|
| GRANTOR           | VARCHAR(128)  |                      | ID de autorización del usuario que ha otorgado los privilegios o SYSIBM.                                                       |
| GRANTEE           | VARCHAR(128)  |                      | ID de autorización del usuario o grupo que tiene los privilegios.                                                              |
| GRANTEETYPE       | CHAR(1)       |                      | U = Se otorga a un usuario individual.<br>G = Se otorga a un grupo.                                                            |
| TABSCHEMA         | VARCHAR(128)  |                      | Nombre calificado de la tabla o vista.                                                                                         |
| TABNAME           | VARCHAR(128)  |                      |                                                                                                                                |
| COLNAME           | VARCHAR(128)  |                      | Nombre de la columna a la que se aplica este privilegio.                                                                       |
| COLNO             | SMALLINT      |                      | Número de esta columna en la tabla o vista.                                                                                    |
| PRIVTYPE          | CHAR(1)       |                      | Indica el tipo de privilegio que se tiene en la tabla o vista:<br>U = Actualiza privilegio<br>R = Hace referencia a privilegio |
| GRANTABLE         | CHAR(1)       |                      | Indica si se puede otorgar el privilegio.<br>G = Otorgable<br>N = No otorgable                                                 |

## SYSCAT.COLCHECKS

---

### SYSCAT.COLCHECKS

Cada fila representa alguna columna a la que una restricción CHECK hace referencia.

Tabla 50. Vista de catálogo SYSCAT.COLCHECKS

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                                                                                                               |
|-------------------|---------------|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CONSTNAME         | VARCHAR(18)   |                      | Nombre de la restricción de comprobación.<br>(Exclusivo en una tabla. Puede ser generado por el sistema.)                                                                                                                                                                                                                                                 |
| TABSCHEMA         | VARCHAR(128)  |                      | Nombre calificado de la tabla que contiene la columna a la que se hace referencia.                                                                                                                                                                                                                                                                        |
| TABNAME           | VARCHAR(128)  |                      |                                                                                                                                                                                                                                                                                                                                                           |
| COLNAME           | VARCHAR(128)  |                      | Nombre de columna.                                                                                                                                                                                                                                                                                                                                        |
| USAGE             | CHAR(1)       |                      | R = Se hace referencia a la columna en la restricción de comprobación.<br>S = La columna es una columna fuente en la restricción de comprobación generada por el sistema que da soporte a una columna generada.<br>T = La columna es una columna destino en la restricción de comprobación generada por el sistema que da soporte a una columna generada. |

## SYSCAT.COLDIST

Contiene estadísticas detalladas de columna para que las utilice el optimizador. Cada fila describe el valor N más frecuente de algunas columnas.

Tabla 51. Vista de catálogo SYSCAT.COLDIST

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                         |
|-------------------|---------------|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TABSCHEMA         | VARCHAR(128)  |                      | Nombre calificado de la tabla a la que se aplica esta entrada.                                                                                                                                                                                      |
| TABNAME           | VARCHAR(128)  |                      |                                                                                                                                                                                                                                                     |
| COLNAME           | VARCHAR(128)  |                      | Nombre de la columna a la que se aplica esta entrada.                                                                                                                                                                                               |
| TYPE              | CHAR(1)       |                      | F = Frecuencia (valor más frecuente)<br>Q = Valor cuantil                                                                                                                                                                                           |
| SEQNO             | SMALLINT      |                      | <ul style="list-style-type: none"> <li>• Si TYPE = F, entonces N en esta columna identifica el valor N más frecuente.</li> <li>• Si TYPE=Q, entonces N en esta columna identifica el valor N cuantil.</li> </ul>                                    |
| COLVALUE          | VARCHAR(254)  | Sí                   | El valor de datos, como un literal de caracteres o un valor nulo.                                                                                                                                                                                   |
| VALCOUNT          | BIGINT        |                      | <ul style="list-style-type: none"> <li>• Si TYPE = F, entonces VALCOUNT es el número de ocurrencias de COLVALUE en la columna.</li> <li>• Si TYPE = Q, entonces VALCOUNT es el número de filas cuyo valor es menor o igual que COLVALUE.</li> </ul> |
| DISTCOUNT         | BIGINT        | Sí                   | Si TYPE = Q, esta columna registra el número de valores diferenciados que son menores o iguales que COLVALUE (nulo si no está disponible).                                                                                                          |

## SYSCAT.COLOPTIONS

---

### SYSCAT.COLOPTIONS

Cada fila contiene los valores de las opciones específicas de columna.

*Tabla 52. Vista de catálogo SYSCAT.COLOPTIONS*

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                  |
|-------------------|---------------|----------------------|------------------------------|
| TABSCHEMA         | VARCHAR(128)  |                      | Calificador de un apodo.     |
| TABNAME           | VARCHAR(128)  |                      | Apodo para la columna.       |
| COLNAME           | VARCHAR(128)  |                      | Nombre de columna local.     |
| OPTION            | VARCHAR(128)  |                      | Nombre de opción de columna. |
| SETTING           | VARCHAR(255)  |                      | Valores                      |

## SYSCAT.COLUMNS

Contiene una fila para cada columna (incluidas las columnas heredadas donde sea aplicable) que se define para una tabla o vista. Todas las vistas de catálogo tienen entradas en la tabla SYSCAT.COLUMNS.

Tabla 53. Vista de catálogo SYSCAT.COLUMNS

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------------------|---------------|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TABSCHEMA         | VARCHAR(128)  |                      | Nombre calificado de la tabla o vista que contiene la columna.                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| TABNAME           | VARCHAR(128)  |                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| COLNAME           | VARCHAR(128)  |                      | Nombre de columna.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| COLNO             | SMALLINT      |                      | Lugar numérico de la columna en la tabla o vista, empezando por cero.                                                                                                                                                                                                                                                                                                                                                                                                                               |
| TYPESHEMA         | VARCHAR(128)  |                      | Contiene el nombre calificado del tipo, si el tipo de datos de la columna es diferenciado. De lo contrario, TYPESHEMA contiene el valor SYSIBM y TYPENAME contiene el tipo de datos de la columna (en formato largo, por ejemplo, CHARACTER). Si se especifica FLOAT o FLOAT( <i>n</i> ) siendo <i>n</i> mayor que 24, TYPENAME se redenomina a DOUBLE. Si se especifica FLOAT( <i>n</i> ) siendo <i>n</i> menor que 25, TYPENAME se redenomina a REAL. También, NUMERIC se redenomina por DECIMAL. |
| TYPENAME          | VARCHAR(18)   |                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| LENGTH            | INTEGER       |                      | Longitud máxima de datos. 0 para tipos diferenciados. La columna LENGTH indica la precisión para los campos DECIMAL.                                                                                                                                                                                                                                                                                                                                                                                |
| SCALE             | SMALLINT      |                      | Escala para los campos DECIMAL; 0 si no es DECIMAL.                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

## SYSCAT.COLUMNS

Tabla 53. Vista de catálogo SYSCAT.COLUMNS (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------------------|---------------|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DEFAULT           | VARCHAR(254)  | Sí                   | <p>El valor por omisión de la columna de una tabla expresado como una constante, un registro especial o una función de conversión adecuada para el tipo de datos de la columna. También puede ser la palabra clave NULL.</p> <p>Los valores se pueden convertir de lo que se ha especificado como valor por omisión. Por ejemplo, las constantes de fecha y hora se presentan en formato ISO y los nombres de función de conversión se califican con el nombre de esquema y los identificadores son delimitados (vea la Nota 3).</p> <p>El valor nulo si no se ha especificado una cláusula DEFAULT o la columna es una columna de vista.</p> |
| NULLS             | CHAR(1)       |                      | <p>Y = La columna puede contener nulos<br/>N = La columna no puede contener nulos.</p> <p>El valor puede ser N para una columna de vista que se deriva de una expresión o función. No obstante, estas columnas permiten nulos cuando la sentencia que utiliza la vista se procesa con avisos para errores aritméticos.</p> <p>Consulte la Nota 1.</p>                                                                                                                                                                                                                                                                                         |
| CODEPAGE          | SMALLINT      |                      | <p>Página de códigos de la columna. Para las columnas de series de caracteres no definidas con el atributo FOR BIT DATA, el valor es la página de códigos de base de datos. Para columnas de series gráficas, el valor es la página de códigos DBCS implícita en la página de códigos de la base de datos (compuesta). De lo contrario, el valor es 0.</p>                                                                                                                                                                                                                                                                                    |



Tabla 53. Vista de catálogo SYSCAT.COLUMNS (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                           |
|-------------------|---------------|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LOGGED            | CHAR(1)       |                      | Sólo se aplica a las columnas cuyo tipo sea LOB o un tipo diferenciado basado en LOB (está en blanco de lo contrario).<br>Y = La columna se anota cronológicamente.<br>N = La columna no se anota cronológicamente.                   |
| COMPACT           | CHAR(1)       |                      | Sólo se aplica a las columnas cuyo tipo sea LOB o un tipo diferenciado basado en LOB (está en blanco de lo contrario).<br>Y = La columna se compacta en el almacenamiento.<br>N = La columna no se compacta.                          |
| COLCARD           | BIGINT        |                      | Número de valores diferenciados en la columna; -1 si no se reúnen estadísticas; -2 para columnas heredadas y columnas de tablas-H.                                                                                                    |
| HIGH2KEY          | VARCHAR(254)  | Sí                   | Segundo nivel más alto de la columna. Este campo está vacío si no se reúnen las estadísticas y para columnas heredadas y columnas de tablas-H. Consulte la nota 2.                                                                    |
| LOW2KEY           | VARCHAR(254)  | Sí                   | Segundo nivel más bajo de la columna. Este campo está vacío si no se reúnen las estadísticas y para columnas heredadas y columnas de tablas-H. Consulte la nota 2.                                                                    |
| AVGCOLLEN         | INTEGER       |                      | Promedio de espacio necesario para la longitud de columna. -1 si es un campo largo o LOB o bien si no se han reunido estadísticas; -2 para columnas heredadas y columnas de tablas-H.                                                 |
| KEYSEQ            | SMALLINT      | Sí                   | La posición numérica de la columna en la clave primaria de la tabla. Este campo es nulo para subtablas y tablas de jerarquía.                                                                                                         |
| PARTKEYSEQ        | SMALLINT      | Sí                   | La posición numérica de la columna en la clave de particionamiento de la tabla. Este campo es nulo ó 0 si la columna no forma parte de la clave de particionamiento. Este campo es también nulo para subtablas y tablas de jerarquía. |

## SYSCAT.COLUMNS

Tabla 53. Vista de catálogo SYSCAT.COLUMNS (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                  |
|-------------------|---------------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NQUANTILES        | SMALLINT      |                      | Número de valores cuantiles registrados en SYSCAT.COLDIST para esta columna; -1 si no se reúnen estadísticas; -2 para columnas heredadas y columnas de tablas-H.                                                                             |
| NMOSTFREQ         | SMALLINT      |                      | Número de los valores más frecuentes registrados en SYSCAT.COLDIST para esta columna; -1 si no se reúnen estadísticas; -2 para columnas heredadas y columnas de tablas-H.                                                                    |
| NUMNULLS          | BIGINT        |                      | Contiene el número de nulos en una columna. -1 si no se reúnen estadísticas.                                                                                                                                                                 |
| TARGET_TYPESCHEMA | VARCHAR(128)  | Sí                   | Nombre calificado del tipo de destino si el tipo de la columna es REFERENCE. Valor nulo si el tipo de la columna no es REFERENCE.                                                                                                            |
| TARGET_TYPENAME   | VARCHAR(18)   | Sí                   | Nombre calificado del tipo de destino si el tipo de la columna es REFERENCE. Valor nulo si el tipo de la columna no es REFERENCE.                                                                                                            |
| SCOPE_TABSCHEMA   | VARCHAR(128)  | Sí                   | Nombre calificado del ámbito (tabla de destino) si el tipo de la columna es REFERENCE. Valor nulo si el tipo de la columna no es REFERENCE o el ámbito no está definido.                                                                     |
| SCOPE_TABNAME     | VARCHAR(128)  | Sí                   | Nombre calificado del ámbito (tabla de destino) si el tipo de la columna es REFERENCE. Valor nulo si el tipo de la columna no es REFERENCE o el ámbito no está definido.                                                                     |
| SOURCE_TABSCHEMA  | VARCHAR(128)  |                      | Nombre calificado de la tabla o vista de la jerarquía respectiva en la que se ha introducido la columna. Para columnas no heredadas, los valores son los mismos que TBCREATOR y TBNAME. Es nulo para columnas de vistas y tablas no de tipo. |
| SOURCE_TABNAME    | VARCHAR(128)  |                      | Nombre calificado de la tabla o vista de la jerarquía respectiva en la que se ha introducido la columna. Para columnas no heredadas, los valores son los mismos que TBCREATOR y TBNAME. Es nulo para columnas de vistas y tablas no de tipo. |

Tabla 53. Vista de catálogo SYSCAT.COLUMNS (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------------------|---------------|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DL_FEATURES       | CHAR(10)      | Sí                   | <p>Sólo se aplica a las columnas de tipo DATALINK. En otro caso, es nulo. Cada posición de carácter se define de la manera siguiente:</p> <ol style="list-style-type: none"> <li>1. Tipo de enlace (U para URL)</li> <li>2. Control de enlace (A para archivo, N para no)</li> <li>3. Integridad (T para todos, N para ninguno)</li> <li>4. Permiso de lectura (S para sistema de archivos, B para base de datos)</li> <li>5. Permiso de grabación (S para sistema de archivos, B para bloqueado)</li> <li>6. Recuperación (S para sí, N para no)</li> <li>7. En desenlace (R para restaurar, S para suprimir, N para no aplicable)</li> </ol> <p>Los caracteres del 8 al 10 se reservan para su utilización en el futuro.</p> |
| SPECIAL_PROPS     | CHAR(8)       | Sí                   | <p>Sólo se aplica a las columnas de tipo REFERENCE. En otro caso, es nulo. Cada posición de carácter se define de la manera siguiente:</p> <p>Columna identificador de objeto (OID) (S para sí, N para no)</p> <p>Generada por el usuario o generada por el sistema (U para usuario, S para sistema)</p>                                                                                                                                                                                                                                                                                                                                                                                                                       |
| HIDDEN            | CHAR(1)       |                      | <p>Tipo de columna oculta</p> <p>S = Columna oculta gestionada por el sistema</p> <p>En blanco si la columna no es oculta</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| INLINE_LENGTH     | INTEGER       |                      | <p>Longitud de la columna de tipo estructurado que se puede mantener con una fila de la tabla base. 0 si no se define explícitamente mediante una sentencia ALTER/CREATE TABLE.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| IDENTITY          | CHAR(1)       |                      | <p>'S' indica que la columna es una columna de identidad; 'N' indica que la columna no es una columna de identidad.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

## SYSCAT.COLUMNS

Tabla 53. Vista de catálogo SYSCAT.COLUMNS (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                              |
|-------------------|---------------|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GENERATED         | CHAR(1)       |                      | Tipo de columna generada<br>A = El valor de la columna es siempre generada<br>D = El valor de la columna se genera por omisión<br>En blanco si la columna no es generada |
| TEXT              | CLOB(64K)     |                      | Contiene el texto de la columna generada, comenzando con la palabra clave AS.                                                                                            |
| REMARKS           | VARCHAR(254)  | Sí                   | Comentario suministrado por el usuario.                                                                                                                                  |

### Nota:

1. Empezando en la Versión 2, el valor D (que indica ningún nulo con valor por omisión) ya no se utiliza. En su lugar, la utilización de WITH DEFAULT se indica por un valor no nulo en la columna DEFAULT.
2. Empezando en la Versión 2, la representación de datos numéricos se ha cambiado a literales de caracteres. El tamaño se ha ampliado de 16 a 33 bytes.
3. Para la Versión 2.1.0, los nombres de función de conversión no se han delimitado y pueden seguir apareciendo de esta manera en la columna DEFAULT. También, algunas columnas de vistas incluyen valores por omisión que todavía aparecen en la columna DEFAULT.

## SYSCAT.CONSTDEP

Contiene una fila para cada dependencia que tiene una restricción de otro objeto.

Tabla 54. Vista de catálogo SYSCAT.CONSTDEP

| Nombre de columna | Tipo de datos | Posibi-<br>lidad de<br>nulos | Descripción                                                                                                                                                 |
|-------------------|---------------|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CONSTNAME         | VARCHAR(18)   |                              | Nombre de la restricción.                                                                                                                                   |
| TABSCHEMA         | VARCHAR(128)  |                              | Nombre calificado de la tabla a la que se aplica la restricción.                                                                                            |
| TABNAME           | VARCHAR(128)  |                              |                                                                                                                                                             |
| BTYPE             | CHAR(1)       |                              | Tipo de objeto del que depende la restricción.<br>Los valores posibles son:<br>F = Instancia de función<br>I = Instancia de índice<br>R = Tipo estructurado |
| BSHEMA            | VARCHAR(128)  |                              | Nombre calificado del objeto del que depende la restricción.                                                                                                |
| BNAME             | VARCHAR(18)   |                              |                                                                                                                                                             |

## SYSCAT.DATATYPES

### SYSCAT.DATATYPES

Contiene una fila para cada tipo de datos, los tipos de datos incorporados y definidos por el usuario inclusive.

Tabla 55. Vista de catálogo SYSCAT.DATATYPES

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                                                                  |
|-------------------|---------------|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TYPESHEMA         | VARCHAR(128)  |                      | Nombre calificado del tipo de datos (para tipos incorporados, TYPESHEMA es SYSIBM).                                                                                                                                                                                                                          |
| TYPENAME          | VARCHAR(18)   |                      |                                                                                                                                                                                                                                                                                                              |
| DEFINER           | VARCHAR(128)  |                      | ID de autorización bajo el cual se ha creado el tipo.                                                                                                                                                                                                                                                        |
| SOURCESHEMA       | VARCHAR(128)  | Sí                   | Nombre calificado del tipo fuente para tipos diferenciados. Nombre calificado del tipo incorporado que se utiliza como el tipo de referencia que sirve de representación para las referencias a tipos estructurados. Nulo para los demás tipos.                                                              |
| SOURCENAME        | VARCHAR(18)   | Sí                   |                                                                                                                                                                                                                                                                                                              |
| METATYPE          | CHAR(1)       |                      | S = Tipo predefinido por el sistema<br>T = Tipo diferenciado<br>R = Tipo estructurado                                                                                                                                                                                                                        |
| TYPEID            | SMALLINT      |                      | Identificador interno del tipo de datos generado por el sistema.                                                                                                                                                                                                                                             |
| SOURCETYPEID      | SMALLINT      | Sí                   | ID de tipo interno del tipo fuente (nulo para tipos incorporados). Para los tipos estructurados definidos por el usuario, éste es el ID de tipo interno del tipo de representación de referencia.                                                                                                            |
| LENGTH            | INTEGER       |                      | Longitud máxima del tipo. 0 para tipos con parámetros predefinidos por el sistema (por ejemplo, DECIMAL y VARCHAR). Para los tipos estructurados definidos por el usuario, indica la longitud del tipo de representación de referencia.                                                                      |
| SCALE             | SMALLINT      |                      | Escala para los tipos diferenciados o tipos de representación de referencia basados en el tipo DECIMAL predefinido por el sistema. 0 para los demás tipos (el propio DECIMAL inclusive). Para los tipos estructurados definidos por el usuario, indica la longitud del tipo de representación de referencia. |

Tabla 55. Vista de catálogo SYSCAT.DATATYPES (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                   |
|-------------------|---------------|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CODEPAGE          | SMALLINT      |                      | Página de códigos para tipos diferenciados de caracteres y gráficos o tipos de representación de referencia; de lo contrario, 0.                                              |
| CREATE_TIME       | TIMESTAMP     |                      | Tiempo de creación del tipo de datos.                                                                                                                                         |
| ATTRCOUNT         | SMALLINT      |                      | Número de atributos en tipo de datos.                                                                                                                                         |
| INSTANTIABLE      | CHAR(1)       |                      | Y = El tipo se puede instanciar.<br>N = El tipo no se puede instanciar.                                                                                                       |
| WITH_FUNC_ACCESS  | CHAR(1)       |                      | Y = Se pueden invocar todos los métodos para este tipo utilizando la función notación.<br>N = No se pueden invocar los métodos para este tipo utilizando la función notación. |
| FINAL             | CHAR(1)       |                      | Y = El tipo definido por el usuario no puede tener subtipos.<br>N = El tipo definido por el usuario puede tener subtipos.                                                     |
| INLINE_LENGTH     | INTEGER       |                      | Longitud de tipo estructurado que se puede mantener con una fila de la tabla base. 0 si no se define explícitamente un valor mediante una sentencia CREATE TYPE.              |
| REMARKS           | VARCHAR(254)  | Sí                   | Comentario suministrado por el usuario o nulo.                                                                                                                                |

## SYSCAT.DBAUTH

---

### SYSCAT.DBAUTH

Registra las autorizaciones de base de datos que ostentan los usuarios.

Tabla 56. Vista de catálogo SYSCAT.DBAUTH

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                      |
|-------------------|---------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GRANTOR           | VARCHAR(128)  |                      | SYSIBM o ID de autorización del usuario que ha otorgado los privilegios.                                                                                         |
| GRANTEE           | VARCHAR(128)  |                      | ID de autorización del usuario o grupo que tiene los privilegios.                                                                                                |
| GRANTEETYPE       | CHAR(1)       |                      | U = Se otorga a un usuario individual.<br>G = Se otorga a un grupo.                                                                                              |
| DBADMAUTH         | CHAR(1)       |                      | Si el destinatario de la operación de otorgar tiene la autorización DBADM sobre la base de datos:<br>Y = Mantiene la autoridad.<br>N = No mantiene la autoridad. |
| CREATETABAUTH     | CHAR(1)       |                      | Si el destinatario de la operación de otorgar puede crear tablas en la base de datos (CREATETAB):<br>Y = Tiene el privilegio.<br>N = No tiene el privilegio.     |
| BINDADDAUTH       | CHAR(1)       |                      | Si el destinatario de la operación de otorgar puede crear paquetes en la base de datos (BINDADD):<br>Y = Tiene el privilegio.<br>N = No tiene el privilegio.     |
| CONNECTAUTH       | CHAR(1)       |                      | Si el destinatario de la operación de otorgar puede conectarse a la base de datos (CONNECT):<br>Y = Tiene el privilegio.<br>N = No tiene el privilegio.          |
| NOFENCEAUTH       | CHAR(1)       |                      | Si el destinatario de la operación de otorgar tiene el privilegio de crear funciones no limitadas.<br>Y = Tiene el privilegio.<br>N = No tiene el privilegio.    |



Tabla 56. Vista de catálogo SYSCAT.DBAUTH (continuación)

| Nombre de columna | Tipo de datos | Posibi-<br>lidad de<br>nulos | Descripción                                                                                                                                                                         |
|-------------------|---------------|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IMPLSCHEMAAUTH    | CHAR(1)       |                              | Si el destinatario de la operación de otorgar puede crear esquemas implícitamente en la base de datos (IMPLICIT_SCHEMA):<br>Y = Tiene el privilegio.<br>N = No tiene el privilegio. |
| LOADAUTH          | CHAR(1)       |                              | Si el usuario autorizado tiene autorización LOAD para la base de datos:<br>Y = Mantiene la autoridad.<br>N = No mantiene la autoridad.                                              |

## SYSCAT.EVENTMONITORS

---

### SYSCAT.EVENTMONITORS

Contiene una fila para cada supervisor de sucesos que se haya definido.

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                               |
|-------------------|---------------|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EVMONNAME         | VARCHAR(18)   |                      | Nombre del supervisor de sucesos.                                                                                                                                                                         |
| DEFINER           | VARCHAR(128)  |                      | ID de autorización del encargado de definir el supervisor de sucesos.                                                                                                                                     |
| TARGET_TYPE       | CHAR(1)       |                      | El tipo del destino en el que se graban los datos del suceso. Valores:<br>F = Archivo<br>P = Conexión                                                                                                     |
| TARGET            | VARCHAR(246)  |                      | Nombre del destino en el que se graban los datos del suceso. Nombre de vía de acceso absoluta del archivo o nombre absoluto de la conexión.                                                               |
| MAXFILES          | INTEGER       | Sí                   | Número máximo de archivos de sucesos que permite este supervisor de sucesos en una vía de acceso de sucesos. Nulo si no hay ningún máximo o si el tipo de destino no es FILE.                             |
| MAXFILESIZE       | INTEGER       | Sí                   | Tamaño máximo (en páginas de 4K) que cada archivo de sucesos puede alcanzar antes de que el supervisor de sucesos cree un nuevo archivo. Nulo si no hay ningún máximo o si el tipo de destino no es FILE. |
| BUFFERSIZE        | INTEGER       | Sí                   | Tamaño de almacenamientos intermedios (en páginas de 4K) utilizadas por los supervisores de sucesos con los destinos de archivo; de lo contrario nulo.                                                    |
| IO_MODE           | CHAR(1)       | Sí                   | Modalidad de E/S de archivo.<br>B = En bloques<br>N = No en bloques<br>Nulo si el tipo de destino no es FILE.                                                                                             |
| WRITE_MODE        | CHAR(1)       | Sí                   | Indica cómo maneja este supervisor los datos de sucesos existentes cuando se activa el supervisor. Valores:<br>A = Añadir<br>R = Sustituir<br>Nulo si el tipo de destino no es FILE.                      |

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                        |
|-------------------|---------------|----------------------|--------------------------------------------------------------------------------------------------------------------|
| AUTOSTART         | CHAR(1)       |                      | El supervisor de sucesos se activará automáticamente cuando se inicie la base de datos.<br>Y = Sí<br>N = No        |
| NODENUM           | SMALLINT      |                      | El número de la partición (o nodo) en que el supervisor de sucesos se ejecuta y anota cronológicamente los sucesos |
| MONSCOPE          | CHAR(1)       |                      | Ámbito de supervisión:<br>L = Local<br>G = Global                                                                  |
| REMARKS           | VARCHAR(254)  | Sí                   | Reservado para su utilización en el futuro.                                                                        |

## SYSCAT.EVENTS

---

### SYSCAT.EVENTS

Contiene una fila para cada suceso que se supervisa. En general, un supervisor de sucesos supervisa varios sucesos.

Tabla 57. Vista de catálogo SYSCAT.EVENTS

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                |
|-------------------|---------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EVMONNAME         | VARCHAR(18)   |                      | Nombre del supervisor de sucesos que supervisa este suceso.                                                                                                |
| TYPE              | VARCHAR(18)   |                      | Tipo del suceso que se supervisa. Los valores posibles son:<br>DATABASE<br>CONNECTIONS<br>TABLES<br>STATEMENTS<br>TRANSACTIONS<br>DEADLOCKS<br>TABLESPACES |
| FILTER            | CLOB(32K)     | Sí                   | Todo el texto de la cláusula WHERE que se aplica a este suceso.                                                                                            |

## SYSCAT.FULLHIERARCHIES

Cada fila representa la relación entre una subtabla y una supertable, un subtipo y un supertipo o una subvista y una supervista. Todas las relaciones jerárquicas, incluyendo las inmediatas, se incluyen en esta vista

Tabla 58. Vista de catálogo SYSCAT.FULLHIERARCHIES

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                              |
|-------------------|---------------|----------------------|--------------------------------------------------------------------------------------------------------------------------|
| METATYPE          | CHAR(1)       |                      | Codifica el tipo de relación:<br>R = Entre tipos estructurados<br>U = Entre tablas con tipo<br>W = Entre vistas con tipo |
| SUB_SCHEMA        | VARCHAR(128)  |                      | Nombre calificado de subtipo, subtabla o subvista.                                                                       |
| SUB_NAME          | VARCHAR(128)  |                      |                                                                                                                          |
| SUPER_SCHEMA      | VARCHAR(128)  |                      | Nombre calificado de supertipo, supertable o supervista.                                                                 |
| SUPER_NAME        | VARCHAR(128)  |                      |                                                                                                                          |
| ROOT_SCHEMA       | VARCHAR(128)  |                      | Nombre calificado de la tabla, vista o tipo que esta en la raíz de la jerarquía.                                         |
| ROOT_NAME         | VARCHAR(128)  |                      |                                                                                                                          |

## SYSCAT.FUNCDEP

### SYSCAT.FUNCDEP

Cada fila representa una dependencia de una función o método respecto de algún otro objeto.

Tabla 59. Vista de catálogo SYSCAT.FUNCDEP

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                                                                                                                                        |
|-------------------|---------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FUNCSCHEMA        | VARCHAR(128)  |                      | Nombre calificado de la función o método que tiene dependencias respecto a otro objeto.                                                                                                                                                                                                                                                                                            |
| FUNCNAME          | VARCHAR(18)   |                      |                                                                                                                                                                                                                                                                                                                                                                                    |
| BTYPE             | CHAR(1)       |                      | Tipo de objeto del que depende la función o método.<br>A = Seudónimo<br>F = Instancia de función o instancia de método<br>O = Dependencia de privilegio para todas las subtablas o subvistas de una jerarquía de tablas o vistas<br>R = Tipo estructurado<br>S = Tabla de resumen<br>T = Tabla<br>U = Tabla con tipo<br>V = Vista<br>W = Vista con tipo<br>X = Extensión de índice |
| BSHEMA            | VARCHAR(128)  |                      | Nombre calificado del objeto del que depende la función o método (si BTYPE='F', este campo es el nombre específico de una función).                                                                                                                                                                                                                                                |
| BNAME             | VARCHAR(128)  |                      |                                                                                                                                                                                                                                                                                                                                                                                    |
| TABAUTH           | SMALLINT      | Sí                   | Si BTYPE = O, S, T, U, V o W, este campo indica los privilegios para la tabla o vista que son necesarios para la función o método dependiente. En otro caso, su valor es nulo.                                                                                                                                                                                                     |

**SYSCAT.FUNCMAPOPTIONS**

Cada fila contiene los valores de las opciones de correlación de funciones.

*Tabla 60. Vista de catálogo SYSCAT.FUNCMAPOPTIONS*

| Nombre de columna | Tipo de datos | Posibi-<br>lidad de<br>nulos | Descripción                                      |
|-------------------|---------------|------------------------------|--------------------------------------------------|
| FUNCTION_MAPPING  | VARCHAR(18)   |                              | Nombre de correlación de funciones.              |
| OPTION            | VARCHAR(128)  |                              | Nombre de la opción de correlación de funciones. |
| SETTING           | VARCHAR(255)  |                              | Valor.                                           |

## SYSCAT.FUNCMAPPARMOPTIONS

---

### SYSCAT.FUNCMAPPARMOPTIONS

Cada fila contiene los valores de las opciones de parámetro de correlación de funciones.

*Tabla 61. Vista de catálogo SYSCAT.FUNCMAPPARMOPTIONS*

| Nombre de columna | Tipo de datos | Posibi-<br>lidad de<br>nulos | Descripción                                                   |
|-------------------|---------------|------------------------------|---------------------------------------------------------------|
| FUNCTION_MAPPING  | VARCHAR(18)   |                              | Nombre de correlación de funciones.                           |
| ORDINAL           | SMALLINT      |                              | Posición de parámetro                                         |
| LOCATION          | CHAR(1)       |                              | L = Local<br>R = Remoto                                       |
| OPTION            | VARCHAR(128)  |                              | Nombre de la opción de parámetro de correlación de funciones. |
| SETTING           | VARCHAR(255)  |                              | Valor.                                                        |



**SYSCAT.FUNCMAPPINGS**

Cada fila contiene las correlaciones de funciones.

Tabla 62. Vista de catálogo SYSCAT.FUNCMAPPINGS

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                             |
|-------------------|---------------|----------------------|-------------------------------------------------------------------------|
| FUNCTION_MAPPING  | VARCHAR(18)   |                      | Nombre de correlación de funciones (puede ser generado por el sistema). |
| FUNCSCHEMA        | VARCHAR(128)  | Sí                   | Esquema de función. Nulo si es una función incorporada del sistema.     |
| FUNCNAME          | VARCHAR(1024) | Sí                   | Nombre de la función local (incorporada o definida por el usuario).     |
| FUNCID            | INTEGER       | Sí                   | Identificador asignado internamente.                                    |
| SPECIFICNAME      | VARCHAR(18)   | Sí                   | Nombre de la instancia de función local.                                |
| DEFINER           | VARCHAR(128)  |                      | ID de autorización bajo el cual se ha creado esta correlación.          |
| WRAPNAME          | VARCHAR(128)  | Sí                   | Nombre de reiniciador al que se aplica esta correlación.                |
| SERVERNAME        | VARCHAR(128)  | Sí                   | Nombre de la fuente de datos.                                           |
| SERVERTYPE        | VARCHAR (30)  | Sí                   | Tipo de fuente de datos a la que se aplica la correlación.              |
| SERVERVERSION     | VARCHAR(18)   | Sí                   | Versión del tipo de servidor al que se aplica la correlación.           |
| CREATE_TIME       | TIMESTAMP     | Sí                   | Hora en que se ha creado la correlación.                                |
| REMARKS           | VARCHAR(254)  | Sí                   | Comentario suministrado por el usuario o nulo.                          |

## SYSCAT.FUNCPARMS

---

### SYSCAT.FUNCPARMS

Contiene una fila para cada parámetro o resultado de una función o método definidos en SYSCAT.FUNCTIONS.

Tabla 63. Vista de catálogo SYSCAT.FUNCPARMS

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                    |
|-------------------|---------------|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FUNCSHEMA         | VARCHAR(128)  |                      | Nombre de función calificado.                                                                                                                                                                                                  |
| FUNCNAME          | VARCHAR(18)   |                      |                                                                                                                                                                                                                                |
| SPECIFICNAME      | VARCHAR(18)   |                      | El nombre de la instancia de función (puede ser generado por el sistema).                                                                                                                                                      |
| ROWTYPE           | CHAR(1)       |                      | P = Parámetro<br>R = Resultado antes de la conversión del tipo de datos<br>C = Resultado después de la conversión del tipo de datos                                                                                            |
| ORDINAL           | SMALLINT      |                      | Si ROWTYPE=P, la posición numérica del parámetro en la signatura de la función. Si ROWTYPE = R y la función devuelve una tabla, la posición numérica de la columna dentro de la tabla resultante. En otro caso, su valor es 0. |
| PARAMNAME         | VARCHAR(128)  |                      | Nombre del parámetro o de la columna del resultado, o nulo si no existe ningún nombre.                                                                                                                                         |
| TYPESHEMA         | VARCHAR(128)  |                      | Nombre calificado de tipo de datos de parámetro o resultado.                                                                                                                                                                   |
| TYPENAME          | VARCHAR(18)   |                      |                                                                                                                                                                                                                                |
| LENGTH            | INTEGER       |                      | Longitud del parámetro o resultado. 0 si el parámetro o resultado es un tipo diferenciado. Consulte la Nota 1.                                                                                                                 |
| SCALE             | SMALLINT      |                      | Escala del parámetro o resultado. 0 si el parámetro o resultado es un tipo diferenciado. Consulte la Nota 1.                                                                                                                   |
| CODEPAGE          | SMALLINT      |                      | Página de códigos del parámetro. 0 indica que no es aplicable o una columna para los datos de caracteres declarada con el atributo FOR BIT DATA                                                                                |
| CAST_FUNCID       | INTEGER       | Sí                   | ID interno de función.                                                                                                                                                                                                         |

Tabla 63. Vista de catálogo SYSCAT.FUNCPARMS (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                     |
|-------------------|---------------|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AS_LOCATOR        | CHAR(1)       |                      | Y = El parámetro o resultado se pasa en forma de un localizador<br>N = No se pasa en forma de un localizador.                                                                                   |
| TARGET_TYPESHEMA  | VARCHAR(128)  |                      | Nombre calificado del tipo de destino si el tipo del parámetro o resultado es REFERENCE. Valor nulo si el tipo del parámetro o resultado no es REFERENCE.                                       |
| TARGET_TYPENAME   | VARCHAR(18)   |                      |                                                                                                                                                                                                 |
| SCOPE_TABSCHEMA   | VARCHAR(128)  |                      | Nombre calificado del ámbito (tipo de destino) si el tipo del parámetro o resultado es REFERENCE. Valor nulo si el tipo del parámetro o resultado no es REFERENCE o el ámbito no está definido. |
| SCOPE_TABNAME     | VARCHAR(128)  |                      |                                                                                                                                                                                                 |
| TRANSFORM_GRPNAME | VARCHAR(18)   | Sí                   | Nombre del grupo transformación para un parámetro de función de tipo estructurado.                                                                                                              |

**Nota:**

1. LENGTH y SCALE se establecen en 0 para las funciones derivadas (funciones definidas con una referencia a otra función) porque heredan la longitud y escala de los parámetros de su fuente.

## SYSCAT.FUNCTIONS

---

### SYSCAT.FUNCTIONS

Contiene una fila para cada función definida por el usuario (escalar, de tabla o fuente), método generado por el sistema o método definido por el usuario. No incluye las funciones incorporadas.

**Nota:** Las descripciones referidas a "funciones" también son aplicables a métodos, a menos que se indique otra cosa.

Tabla 64. Vista de catálogo SYSCAT.FUNCTIONS

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                     |
|-------------------|---------------|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FUNCSHEMA         | VARCHAR(128)  |                      | Nombre de función calificado.                                                                                                                                                   |
| FUNCNAME          | VARCHAR(18)   |                      |                                                                                                                                                                                 |
| SPECIFICNAME      | VARCHAR(18)   |                      | El nombre de la instancia de función (puede ser generado por el sistema).                                                                                                       |
| DEFINER           | VARCHAR(128)  |                      | ID de autorización del encargado de definir la función.                                                                                                                         |
| FUNCID            | INTEGER       |                      | ID de función asignado internamente.                                                                                                                                            |
| RETURN_TYPE       | SMALLINT      |                      | Tipo de código de retorno de tipo interno de función.                                                                                                                           |
| ORIGIN            | CHAR(1)       |                      | B = Incorporado<br>E = Definido por el usuario, externo<br>Q = Definido por el usuario, SQL<br>U = Definido por el usuario, basado en una fuente<br>S = Generado por el sistema |
| TYPE              | CHAR(1)       |                      | C = Función de columna<br>R = Función de fila<br>S = Función escalar<br>T = Función de tabla                                                                                    |
| METHOD            | CHAR(1)       |                      | Y = Método<br>N = No es un método                                                                                                                                               |
| EFFECT            | CHAR(2)       |                      | MU = método mutador<br>OB = método observador<br>CN = método constructor<br>Blancos = No es un método generado por el sistema                                                   |

Tabla 64. Vista de catálogo SYSCAT.FUNCTIONS (continuación)

| Nombre de columna | Tipo de datos                | Posibilidad de nulos | Descripción                                                                                                                                                                                                                  |
|-------------------|------------------------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PARAM_COUNT       | SMALLINT                     |                      | Número de parámetros de función.                                                                                                                                                                                             |
| PARAM_SIGNATURE   | VARCHAR(180)<br>FOR BIT DATA |                      | Concatenación de un máximo de 90 tipos de parámetros, en formato interno. Longitud cero si la función no toma parámetros.                                                                                                    |
| CREATE_TIME       | TIMESTAMP                    |                      | Indicación de la hora de la creación de la función. Se establece en 0 para funciones de la Versión 1.                                                                                                                        |
| QUALIFIER         | VARCHAR(128)                 |                      | Valor del esquema por omisión en el momento de definición de objeto.                                                                                                                                                         |
| WITH_FUNC_ACCESS  | CHAR(1)                      |                      | Y = El método se puede invocar utilizando una notación funcional<br>N = El método no se puede invocar utilizando una notación funcional                                                                                      |
| TYPE_PRESERVING   | CHAR(1)                      |                      | Y = El tipo de retorno está determinado por un parámetro que conserva el tipo. Todos los métodos mutadores generados por el sistema conservan el tipo.<br>N = El tipo de retorno es el tipo de retorno declarado del método. |
| VARIANT           | CHAR(1)                      |                      | Y = Variante (los resultados pueden diferir)<br>N = No variante (los resultados son coherentes)<br>En blanco si ORIGIN no es E                                                                                               |
| SIDE_EFFECTS      | CHAR(1)                      |                      | E = La función tiene efectos adicionales externos (el número de invocaciones es importante)<br>N = Ningún efecto adicional<br>En blanco si ORIGIN no es E                                                                    |
| FENCED            | CHAR(1)                      |                      | Y = Limitado<br>N = No limitado<br>En blanco si ORIGIN no es E                                                                                                                                                               |

## SYSCAT.FUNCTIONS

Tabla 64. Vista de catálogo SYSCAT.FUNCTIONS (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                    |
|-------------------|---------------|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NULLCALL          | CHAR(1)       |                      | Y = CALLED ON NULL INPUT<br>N = RETURNS NULL ON NULL INPUT<br>(el resultado de la función es implícitamente nulo si el(los) operando(s) es(son) nulo(s).<br>En blanco si ORIGIN no es E.                                       |
| CAST_FUNCTION     | CHAR(1)       |                      | Y = Es una función de conversión de tipos de datos<br>N = No es una función de conversión de tipos de datos                                                                                                                    |
| ASSIGN_FUNCTION   | CHAR(1)       |                      | Y = Función de asignación implícita<br>N = No es una función de asignación                                                                                                                                                     |
| SCRATCHPAD        | CHAR(1)       |                      | Y = Esta función tiene una memoria de trabajo.<br>N = Esta función no tiene una memoria de trabajo.<br>En blanco si ORIGIN no es E                                                                                             |
| FINAL_CALL        | CHAR(1)       |                      | Y = Se realiza una llamada final a esta función en el fin de sentencia.<br>N = No se realiza ninguna llamada final.<br>En blanco si ORIGIN no es E                                                                             |
| PARALLELIZABLE    | CHAR(1)       |                      | Y = La función se puede ejecutar en paralelo.<br>N = La función no se puede ejecutar en paralelo.<br>En blanco si ORIGIN no es E                                                                                               |
| CONTAINS_SQL      | CHAR(1)       |                      | Indica si una función o método contiene código SQL.<br>C = CONTAINS SQL: sólo se permite SQL que no lee ni modifica datos SQL.<br>N = NO SQL: no se permite SQL.<br>R = READS SQL DATA: sólo se permite SQL que lee datos SQL. |

Tabla 64. Vista de catálogo SYSCAT.FUNCTIONS (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                       |
|-------------------|---------------|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DBINFO            | CHAR(1)       |                      | Indica si se pasa un parámetro DBINFO a una función externa.<br>Y = DBINFO se pasa.<br>N = DBINFO no se pasa.<br>En blanco si ORIGIN no es E                                                                                      |
| RESULT_COLS       | SMALLINT      |                      | Para una función de tabla (TYPE=T) contiene el número de columnas de la tabla resultante; de lo contrario contiene 1.                                                                                                             |
| LANGUAGE          | CHAR(8)       |                      | La implantación del cuerpo del lenguaje de función. Posibles valores son C, JAVA, OLE u OLEDB. En blanco si ORIGIN no es E ni Q.                                                                                                  |
| IMPLEMENTATION    | VARCHAR(254)  | Sí                   | Si ORIGIN = E, identifica la vía de acceso/módulo/función que implanta esta función. Si ORIGIN = U y la función fuente está incorporada, esta columna contiene el nombre y signatura de la función fuente. En otro caso, es nulo. |
| CLASS             | VARCHAR(128)  | Sí                   | Si LANGUAGE = JAVA, identifica la clase que implanta esta función. En otro caso, es nulo.                                                                                                                                         |
| JAR_ID            | VARCHAR(128)  | Sí                   | Si LANGUAGE = JAVA, identifica el archivo jar que implanta esta función. En otro caso, es nulo.                                                                                                                                   |
| PARAM_STYLE       | CHAR(8)       |                      | Indica el estilo de parámetro declarado en la sentencia CREATE FUNCTION. Valores:<br>DB2SQL<br>DB2GENRL<br>JAVA<br>En blanco si ORIGIN no es E                                                                                    |
| SOURCE_SCHEMA     | VARCHAR(128)  | Sí                   | Si ORIGIN = U y la función fuente es una función definida por el usuario, contiene el nombre calificado de la función fuente. Si                                                                                                  |
| SOURCE_SPECIFIC   | VARCHAR(18)   | Sí                   | ORIGIN = U y la función fuente es incorporada, SOURCE_SCHEMA es 'SYSIBM' y SOURCE_SPECIFIC es 'N/D para incorporada'. Nulo si ORIGIN no es U.                                                                                     |
| IOS_PER_INVOC     | DOUBLE        |                      | El número estimado de E/S por invocación; -1 si no se conoce (0 por omisión).                                                                                                                                                     |

## SYSCAT.FUNCTIONS

Tabla 64. Vista de catálogo SYSCAT.FUNCTIONS (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                          |
|-------------------|---------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| INSTS_PER_INVOC   | DOUBLE        |                      | El número estimado de instrucciones por invocación; -1 si no se conoce (450 por omisión).                                                                                                                                                            |
| IOS_PER_ARGBYTE   | DOUBLE        |                      | Número estimado de E/S por byte de argumento de entrada; -1 si no se conoce (0 por omisión).                                                                                                                                                         |
| INSTS_PER_ARGBYTE | DOUBLE        |                      | Número estimado de instrucciones por byte de argumento de entrada; -1 si no se conoce (0 por omisión).                                                                                                                                               |
| PERCENT_ARGBYTES  | SMALLINT      |                      | Porcentaje promedio estimado de bytes de argumento de entrada que leerá la función realmente; -1 si no se conoce (100 por omisión).                                                                                                                  |
| INITIAL_IOS       | DOUBLE        |                      | Número estimado de E/S realizadas la primera/última vez que se invoca la función; -1 si no se conoce (0 por omisión).                                                                                                                                |
| INITIAL_INSTS     | DOUBLE        |                      | Número estimado de instrucciones ejecutadas la primera/última vez que se invoca la función; -1 si no se conoce (0 por omisión).                                                                                                                      |
| CARDINALITY       | BIGINT        |                      | La cardinalidad prevista de una función de tabla. -1 si no se conoce o si la función es una función de tabla.                                                                                                                                        |
| IMPLEMENTED       | CHAR(1)       |                      | Y = la función está implementada.<br>M = el método está implementado y no tiene acceso a función. Vea la nota 1.<br>H = el método está implementado y tiene acceso a función. Vea la nota 1.<br>N = especificación de método sin una implementación. |
| SELECTIVITY       | DOUBLE        |                      | Se utiliza para predicados definidos por el usuario. -1 si no hay ningún predicado definido por el usuario. Consulte la nota 2.                                                                                                                      |
| OVERRIDEN_FUNCID  | INTEGER       | Sí                   | Reservado para su utilización en el futuro.                                                                                                                                                                                                          |
| SUBJECT_TYPESHEMA | VARCHAR(128)  | Sí                   | Esquema de tipo sujeto para el método definido por el usuario.                                                                                                                                                                                       |
| SUBJECT_TYPENAME  | VARCHAR(18)   | Sí                   | Nombre de tipo sujeto para el método definido por el usuario.                                                                                                                                                                                        |



Tabla 64. Vista de catálogo SYSCAT.FUNCTIONS (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                      |
|-------------------|---------------|----------------------|----------------------------------------------------------------------------------|
| FUNC_PATH         | VARCHAR(254)  | Sí                   | Vía de acceso de función en el momento en que se ha definido la función.         |
| BODY              | CLOB(1M)      | Sí                   | Si el lenguaje es SQL, el texto de la sentencia CREATE FUNCTION o CREATE METHOD. |
| REMARKS           | VARCHAR(254)  | Sí                   | Comentario suministrado por el usuario o nulo.                                   |

**Nota:**

1. Este valor puede que no aparezca en versiones futuras de DB2.
2. Esta columna se establece en -1 durante la migración en el descriptor empaquetado y catálogos del sistema para todas las funciones definidas por el usuario. Para un predicado definido por el usuario, la selectividad es -1 en el catálogo del sistema. En este caso, el valor de selectividad utilizado por el optimizador es 0.01.

## SYSCAT.HIERARCHIES

---

### SYSCAT.HIERARCHIES

Cada fila representa la relación entre una subtabla y su supertabla inmediata, un subtipo y su supertipo inmediato o una subvista y su supervista inmediata. Sólo las relaciones jerárquicas inmediatas se incluyen en esta vista.

Tabla 65. Vista de catálogo SYSCAT.HIERARCHIES

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                              |
|-------------------|---------------|----------------------|--------------------------------------------------------------------------------------------------------------------------|
| METATYPE          | CHAR(1)       |                      | Codifica el tipo de relación:<br>R = Entre tipos estructurados<br>U = Entre tablas con tipo<br>W = Entre vistas con tipo |
| SUB_SCHEMA        | VARCHAR(128)  |                      | Nombre calificado de subtipo, subtabla o subvista.                                                                       |
| SUB_NAME          | VARCHAR(128)  |                      |                                                                                                                          |
| SUPER_SCHEMA      | VARCHAR(128)  |                      | Nombre calificado de supertipo, supertabla o supervista.                                                                 |
| SUPER_NAME        | VARCHAR(128)  |                      |                                                                                                                          |
| ROOT_SCHEMA       | VARCHAR(128)  |                      | Nombre calificado de la tabla, vista o tipo que esta en la raíz de la jerarquía.                                         |
| ROOT_NAME         | VARCHAR(128)  |                      |                                                                                                                          |

## SYSCAT.INDEXAUTH

Contiene una fila para cada privilegio que se posea en un índice.

Tabla 66. Vista de catálogo SYSCAT.INDEXAUTH

| Nombre de columna | Tipo de datos | Posibi-<br>lidad de<br>nulos | Descripción                                                                                                                                        |
|-------------------|---------------|------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| GRANTOR           | VARCHAR(128)  |                              | ID de autorización del usuario que ha otorgado los privilegios.                                                                                    |
| GRANTEE           | VARCHAR(128)  |                              | ID de autorización del usuario o grupo que tiene los privilegios.                                                                                  |
| GRANTEETYPE       | CHAR(1)       |                              | U = Se otorga a un usuario individual.<br>G = Se otorga a un grupo.                                                                                |
| INDSCHEMA         | VARCHAR(128)  |                              | Nombre del índice.                                                                                                                                 |
| INDNAME           | VARCHAR(18)   |                              |                                                                                                                                                    |
| CONTROLAUTH       | CHAR(1)       |                              | Si el destinatario de la operación de otorgar posee el privilegio CONTROL en el índice:<br>Y = Tiene el privilegio.<br>N = No tiene el privilegio. |

## SYSCAT.INDEXCOLUSE

---

### SYSCAT.INDEXCOLUSE

Lista todas las columnas que participan en un índice.

*Tabla 67. Vista de catálogo SYSCAT.INDEXCOLUSE*

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                         |
|-------------------|---------------|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| INDSCHEMA         | VARCHAR(128)  |                      | Nombre calificado del índice.                                                                                                                       |
| INDNAME           | VARCHAR(18)   |                      |                                                                                                                                                     |
| COLNAME           | VARCHAR(128)  |                      | Nombre de la columna.                                                                                                                               |
| COLSEQ            | SMALLINT      |                      | Posición numérica de la columna en el índice (posición inicial = 1).                                                                                |
| COLORDER          | CHAR(1)       |                      | Orden de los valores de esta columna en el índice. Valores:<br>A = Ascendente<br>D = Descendente<br>I = Columna INCLUDE (se pasa por alto el orden) |

## SYSCAT.INDEXDEP

Cada fila representa una dependencia de un índice respecto de algún otro objeto.

Tabla 68. Vista de catálogo SYSCAT.INDEXDEP

| Nombre de columna | Tipo de datos | Posibi-<br>lidad de<br>nulos | Descripción                                                                                                                                                                                                                                                                                                                                           |
|-------------------|---------------|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| INDSCHEMA         | VARCHAR(128)  |                              | Nombre calificado del índice que depende de otro objeto.                                                                                                                                                                                                                                                                                              |
| INDNAME           | VARCHAR(18)   |                              |                                                                                                                                                                                                                                                                                                                                                       |
| BTYPE             | CHAR(1)       |                              | El tipo de objeto del que depende el índice.<br>A = Seudónimo<br>F = Instancia de función<br>O = Dependencia de privilegio para todas las subtablas o subvistas de una jerarquía de tablas o vistas<br>R = Tipo estructurado<br>S = Tabla de resumen<br>T = Tabla<br>U = Tabla con tipo<br>V = Vista<br>W = Vista con tipo<br>X = Extensión de índice |
| BSCHEMA           | VARCHAR(128)  |                              | Nombre calificado del objeto del que depende el índice.                                                                                                                                                                                                                                                                                               |
| BNAME             | VARCHAR(128)  |                              |                                                                                                                                                                                                                                                                                                                                                       |
| TABAUTH           | SMALLINT      | Sí                           | Si BTYPE = O, S, T, U, V o W, codifica los privilegios de la tabla o la vista que son necesarios para el índice dependiente. De lo contrario es nulo.                                                                                                                                                                                                 |

## SYSCAT.INDEXES

---

### SYSCAT.INDEXES

Contiene una fila para cada índice (incluidos los índices heredados donde sea aplicable) que se define para una tabla.

Tabla 69. Vista de catálogo SYSCAT.INDEXES

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                                                |
|-------------------|---------------|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| INDSCHEMA         | VARCHAR(128)  |                      | Nombre del índice.                                                                                                                                                                                                                                                                         |
| INDNAME           | VARCHAR(18)   |                      |                                                                                                                                                                                                                                                                                            |
| DEFINER           | VARCHAR(128)  |                      | Usuario que ha creado el índice.                                                                                                                                                                                                                                                           |
| TABSCHEMA         | VARCHAR(128)  |                      | Nombre calificado de la tabla o apodo en el que se define el índice.                                                                                                                                                                                                                       |
| TABNAME           | VARCHAR(128)  |                      |                                                                                                                                                                                                                                                                                            |
| COLNAMES          | VARCHAR(640)  |                      | Lista de nombres de columna, cada uno precedido por un signo + o – para indicar orden ascendente o descendente respectivamente.<br>Aviso: Esta columna se eliminará en el futuro.<br>Utilice “SYSCAT.INDEXCOLUSE” en la página 1326 para esta información.                                 |
| UNIQUERULE        | CHAR(1)       |                      | Regla de unicidad:<br>D = Los duplicados están permitidos<br>P = Índice primario<br>U = Sólo se permiten entradas exclusivas                                                                                                                                                               |
| MADE_UNIQUE       | CHAR(1)       |                      | Y = El índice era de no unicidad originalmente, pero se ha convertido en un índice de unicidad para dar soporte a una restricción de clave primaria o de unicidad. Si se elimina la restricción, el índice volverá a ser de no unicidad.<br>N = El índice permanece tal como se ha creado. |
| COLCOUNT          | SMALLINT      |                      | Número de columnas de la clave más el número de columnas INCLUDE si hay alguna.                                                                                                                                                                                                            |
| UNIQUE_COLCOUNT   | SMALLINT      |                      | Número de columnas necesarias para una clave de unicidad. Siempre <=COLCOUNT. < COLCOUNT solamente si hay columnas INCLUDE. -1 si el índice no tiene clave de unicidad (permite duplicados)                                                                                                |
| INDEXTYPE         | CHAR(4)       |                      | Tipo de índice.<br>CLUS = Agrupación<br>REG = Normal                                                                                                                                                                                                                                       |

Tabla 69. Vista de catálogo SYSCAT.INDEXES (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                       |
|-------------------|---------------|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ENTRYTYPE         | CHAR(1)       |                      | H = Un índice en una tabla de jerarquía (tabla-H)<br>L = Índice lógico en una tabla con tipo en blanco si es un índice en una tabla de no tipo                                                    |
| PCTFREE           | SMALLINT      |                      | Porcentaje de cada página de hoja de índice que se va a reservar durante la creación inicial del índice. Este espacio está disponible para inserciones futuras después de la creación del índice. |
| IID               | SMALLINT      |                      | ID interno de índice.                                                                                                                                                                             |
| NLEAF             | INTEGER       |                      | Número de páginas; -1 si no se reúnen estadísticas.                                                                                                                                               |
| NLEVELS           | SMALLINT      |                      | Número de niveles de índices; -1 si no se reúnen estadísticas.                                                                                                                                    |
| FIRSTKEYCARD      | BIGINT        |                      | Número de valores de primera clave diferenciada; -1 si no se reúnen estadísticas.                                                                                                                 |
| FIRST2KEYCARD     | BIGINT        |                      | Número de claves diferenciadas que utilizan las dos primeras columnas del índice (-1 si no hay estadísticas o no son aplicables)                                                                  |
| FIRST3KEYCARD     | BIGINT        |                      | Número de claves diferenciadas que utilizan las tres primeras columnas del índice (-1 si no hay estadísticas o si no son aplicables)                                                              |
| FIRST4KEYCARD     | BIGINT        |                      | Número de claves diferenciadas que utilizan las cuatro primeras columnas del índice (-1 si no hay estadísticas o no son aplicables)                                                               |
| FULLKEYCARD       | BIGINT        |                      | Número de valores de clave completa diferenciada; -1 si no se reúnen estadísticas.                                                                                                                |
| CLUSTERRATIO      | SMALLINT      |                      | Grado de agrupación de los datos con el índice; -1 si no se reúnen estadísticas o si se reúnen estadísticas de índice detalladas (en cuyo caso, se utilizará CLUSTERFACTOR en su lugar).          |
| CLUSTERFACTOR     | DOUBLE        |                      | Mejor medición del grado de agrupación, o -1 si no se han reunido estadísticas de índice detalladas o si el índice está definido en un apodo.                                                     |

## SYSCAT.INDEXES

Tabla 69. Vista de catálogo SYSCAT.INDEXES (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                                                                                                                                                       |
|-------------------|---------------|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SEQUENTIAL_PAGES  | INTEGER       |                      | Número de páginas ubicadas en disco por orden de clave de índice con pocos o ningún vacío entre ellas. (-1 si no hay estadísticas disponibles.)                                                                                                                                                                                                                                                   |
| DENSITY           | INTEGER       |                      | Proporción de SEQUENTIAL_PAGES para numerar las páginas del rango de páginas ocupadas por el índice, expresada como un porcentaje (entero entre 0 y 100, -1 si no hay estadísticas disponibles.)                                                                                                                                                                                                  |
| USER_DEFINED      | SMALLINT      |                      | 1 si un usuario ha definido este índice y no se ha eliminado; de lo contrario, 0.                                                                                                                                                                                                                                                                                                                 |
| SYSTEM_REQUIRED   | SMALLINT      |                      | 1 si este índice es necesario para la restricción de clave primaria o de clave de unicidad O BIEN si es el índice en la columna de identificador de objeto (OID) de una tabla con tipo.<br>2 si este índice es necesario para la restricción de clave primaria o de clave de unicidad Y es el índice en la columna de identificador de objeto (OID) de una tabla con tipo.<br>De lo contrario, 0. |
| CREATE_TIME       | TIMESTAMP     |                      | Hora en que se ha creado el índice.                                                                                                                                                                                                                                                                                                                                                               |
| STATS_TIME        | TIMESTAMP     | Sí                   | Última vez que se ha realizado un cambio en las estadísticas registradas para este índice. Nulo si no hay estadísticas disponibles.                                                                                                                                                                                                                                                               |
| PAGE_FETCH_PAIRS  | VARCHAR(254)  |                      | Una lista de pares de enteros, representada en la forma de caracteres. Cada par representa el número de páginas de un almacenamiento intermedio hipotético y el número de lecturas de páginas necesario para explorar la tabla con este índice utilizando dicho almacenamiento intermedio hipotético. (Serie de longitud cero si no hay datos disponibles.)                                       |
| MINPCTUSED        | SMALLINT      |                      | Si no es cero, se habilita la reorganización del índice en línea y el valor es el umbral del espacio mínimo utilizado antes de fusionar las páginas.                                                                                                                                                                                                                                              |



Tabla 69. Vista de catálogo SYSCAT.INDEXES (continuación)

| Nombre de columna | Tipo de datos | Posibi-<br>lidad de<br>nulos | Descripción                                                                                               |
|-------------------|---------------|------------------------------|-----------------------------------------------------------------------------------------------------------|
| REVERSE_SCANS     | CHAR(1)       |                              | Y = El índice soporta exploraciones<br>invertidas<br>N = El índice no soporta exploraciones<br>invertidas |
| INTERNAL_FORMAT   | SMALLINT      |                              | Codifica la representación interna del índice.                                                            |
| REMARKS           | VARCHAR(254)  | Sí                           | Comentario suministrado por el usuario o nulo.                                                            |

## SYSCAT.INDEXOPTIONS

---

### SYSCAT.INDEXOPTIONS

Cada fila contiene los valores de las opciones específicas de índice.

*Tabla 70. Vista de catálogo SYSCAT.INDEXOPTIONS*

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                    |
|-------------------|---------------|----------------------|--------------------------------|
| INDSHEMA          | VARCHAR(128)  |                      | Nombre de esquema del índice.  |
| INDNAME           | VARCHAR(18)   |                      | Nombre local del índice.       |
| OPTION            | VARCHAR(128)  |                      | Nombre de la opción de índice. |
| SETTING           | VARCHAR(255)  |                      | Valor.                         |

---

**SYSCAT.KEYCOLUSE**

Lista todas las columnas que participan en una clave (incluidas las claves primarias o de unicidad heredadas donde sea aplicable) definidas por una restricción de unicidad, de clave primaria o de clave foránea.

*Tabla 71. Vista de catálogo SYSCAT.KEYCOLUSE*

| Nombre de columna | Tipo de datos | Posibi-<br>lidad de<br>nulos | Descripción                                                       |
|-------------------|---------------|------------------------------|-------------------------------------------------------------------|
| CONSTNAME         | VARCHAR(18)   |                              | Nombre de la restricción (exclusivo en una tabla).                |
| TABSCHEMA         | VARCHAR(128)  |                              | Nombre calificado de la tabla que contiene la columna.            |
| TABNAME           | VARCHAR(128)  |                              |                                                                   |
| COLNAME           | VARCHAR(128)  |                              | Nombre de la columna.                                             |
| COLSEQ            | SMALLINT      |                              | Posición numérica de la columna en la clave (posición inicial=1). |

## SYSCAT.NAMEMAPPINGS

---

### SYSCAT.NAMEMAPPINGS

Cada fila representa la correlación entre los objetos lógicos y los objetos de implantación correspondientes que implantan los objetos lógicos.

Tabla 72. Vista de catálogo SYSCAT.NAMEMAPPINGS

| Nombre de columna | Tipo de datos | Posibi-<br>lidad de<br>nulos | Descripción                                                                       |
|-------------------|---------------|------------------------------|-----------------------------------------------------------------------------------|
| TYPE              | CHAR(1)       |                              | C = Columna<br>I = Índice<br>U = Tabla con tipo                                   |
| LOGICAL_SCHEMA    | VARCHAR(128)  |                              | Nombre calificado del objeto lógico.                                              |
| LOGICAL_NAME      | VARCHAR(128)  |                              |                                                                                   |
| LOGICAL_COLNAME   | VARCHAR(128)  | Sí                           | Si TYPE = C, el nombre de la columna lógica.<br>De lo contrario es nulo.          |
| IMPL_SCHEMA       | VARCHAR(128)  |                              | Nombre calificado del objeto de implantación<br>que implanta el objeto lógico.    |
| IMPL_NAME         | VARCHAR(128)  |                              |                                                                                   |
| IMPL_COLNAME      | VARCHAR(128)  | Sí                           | Si TYPE = C, el nombre de la columna de<br>implantación. De lo contrario es nulo. |

## SYSCAT.NODEGROUPDEF

Contiene una fila para cada partición que esté contenida en un grupo de nodos.

Tabla 73. Vista de catálogo SYSCAT.NODEGROUPDEF

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-------------------|---------------|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NGNAME            | VARCHAR(18)   |                      | El nombre del grupo de nodos que contiene la partición (o nodo).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| NODENUM           | SMALLINT      |                      | El número de partición (o nodo) de una partición contenida en el grupo de nodos. Un número de partición válido está entre 0 y 999 inclusive.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| IN_USE            | CHAR(1)       |                      | Estado de la partición (o nodo).<br><p>A = La partición recién añadida no está en el mapa de particionamiento, pero se crean los contenedores para los espacios de tablas en el grupo de nodos. La partición se añade al mapa de particionamiento cuando se completa satisfactoriamente una operación Redistribuir grupo de nodos.</p> <p>D = La partición se eliminará cuando se complete una operación Redistribuir grupo de nodos.</p> <p>T = La partición que se acaba de añadir no está en el mapa de particionamiento y se ha añadido utilizando la cláusula WITHOUT TABLESPACES. Los contenedores deben añadirse específicamente a los espacios de tablas para el grupo de nodos.</p> <p>Y = La partición está en el mapa de particionamiento.</p> |

## SYSCAT.NODEGROUPS

---

### SYSCAT.NODEGROUPS

Contiene una fila para cada grupo de nodos.

Tabla 74. Vista de catálogo SYSCAT.NODEGROUPS

| Nombre de columna | Tipo de datos | Posibi-<br>lidad de<br>nulos | Descripción                                                                                                                                                       |
|-------------------|---------------|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NGNAME            | VARCHAR(18)   |                              | Nombre del grupo de nodos.                                                                                                                                        |
| DEFINER           | VARCHAR(128)  |                              | ID de autorización del encargado de definir el grupo de nodos.                                                                                                    |
| PMAP_ID           | SMALLINT      |                              | Identificador del mapa de particionamiento en SYSCAT.PARTITIONMAPS.                                                                                               |
| REBALANCE_PMAP_ID | SMALLINT      |                              | Identificador del mapa de particionamiento que se utiliza actualmente para la redistribución. El valor es -1 si la redistribución no está en proceso actualmente. |
| CREATE_TIME       | TIMESTAMP     |                              | Hora de creación del grupo de nodos.                                                                                                                              |
| REMARKS           | VARCHAR(254)  | Sí                           | Comentario proporcionado por el usuario.                                                                                                                          |

## SYSCAT.PACKAGEAUTH

Contiene una fila para cada privilegio que se posea en un paquete.

Tabla 75. Vista de catálogo SYSCAT.PACKAGEAUTH

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                |
|-------------------|---------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GRANTOR           | VARCHAR(128)  |                      | ID de autorización del usuario que ha otorgado los privilegios.                                                                                            |
| GRANTEE           | VARCHAR(128)  |                      | ID de autorización del usuario o grupo que tiene los privilegios.                                                                                          |
| GRANTEETYPE       | CHAR(1)       |                      | U = Se otorga a un usuario individual.<br>G = Se otorga a un grupo.                                                                                        |
| PKGSHEMA          | VARCHAR(128)  |                      | Nombre del paquete en el que se tienen los privilegios.                                                                                                    |
| PKGNAME           | CHAR(8)       |                      |                                                                                                                                                            |
| CONTROLAUTH       | CHAR(1)       |                      | Indica si el destinatario de la operación de otorgar posee el privilegio CONTROL en el paquete:<br>Y = Tiene el privilegio.<br>N = No tiene el privilegio. |
| BINDAUTH          | CHAR(1)       |                      | Indica si el destinatario de la operación de otorgar posee el privilegio BIND en el paquete:<br>Y = Tiene el privilegio.<br>N = No tiene el privilegio.    |
| EXECUTEAUTH       | CHAR(1)       |                      | Indica si el destinatario de la operación de otorgar posee el privilegio EXECUTE en el paquete:<br>Y = Tiene el privilegio.<br>N = No tiene el privilegio. |

## SYSCAT.PACKAGEDEP

---

### SYSCAT.PACKAGEDEP

Contiene una fila para cada dependencia que tienen los paquetes de índices, tablas, vistas, funciones, seudónimos, tipos y jerarquías.

Tabla 76. Vista de catálogo SYSCAT.PACKAGEDEP

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                                                                                                                                                                           |
|-------------------|---------------|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PKGSHEMA          | VARCHAR(128)  |                      | Nombre del paquete.                                                                                                                                                                                                                                                                                                                                                                                                   |
| PKGNAME           | CHAR(8)       |                      |                                                                                                                                                                                                                                                                                                                                                                                                                       |
| BINDER            | VARCHAR(128)  | Sí                   | Enlazador del paquete.                                                                                                                                                                                                                                                                                                                                                                                                |
| BTYPE             | CHAR(1)       |                      | Tipo de objeto BNAME:<br>A = Seudónimo<br>D = Definición de servidor<br>F = Instancia de función<br>I = Índice<br>M = Correlación de funciones<br>N = Apodo<br>O = Dependencia de privilegio de todas las subtablas o subvistas de una jerarquía de tablas o de vistas<br>P = Tamaño de página<br>R = Tipo estructurado<br>S = Tabla de resumen<br>T = Tabla<br>U = Tabla con tipo<br>V = Vista<br>W = Vista con tipo |
| BSHEMA            | VARCHAR(128)  |                      | Nombre calificado de un objeto del que depende el paquete.                                                                                                                                                                                                                                                                                                                                                            |
| BNAME             | VARCHAR(128)  |                      |                                                                                                                                                                                                                                                                                                                                                                                                                       |
| TABAUTH           | SMALLINT      | Sí                   | Si BTYPE es O, S, T, U, V o W cuando codifica los privilegios que necesita este paquete (Select, Insert, Delete, Update).                                                                                                                                                                                                                                                                                             |

**Nota:** Cuando se elimina una instancia de función de la que depende un paquete, éste se coloca en un estado “no operativo” a partir del cual se ha de volver a enlazar explícitamente. Cuando se elimina cualquier otro objeto del que depende el paquete, éste se coloca en un estado “no válido” a partir del cual el sistema intentará volverlo a enlazar automáticamente la primera vez que se haga referencia al paquete.



## SYSCAT.PACKAGES

Contiene una fila para cada paquete que se ha creado mediante el enlace lógico de un programa de aplicación.

Tabla 77. Vista de catálogo SYSCAT.PACKAGES

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                 |
|-------------------|---------------|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PKGSHEMA          | VARCHAR(128)  |                      | Nombre del paquete.                                                                                                                                                                                                                                         |
| PKGNAME           | CHAR(8)       |                      |                                                                                                                                                                                                                                                             |
| BOUNDBY           | VARCHAR(128)  |                      | ID de autorización (OWNER) del enlazador del paquete.                                                                                                                                                                                                       |
| DEFINER           | VARCHAR(128)  |                      | ID de usuario bajo el cual se ha enlazado el paquete.                                                                                                                                                                                                       |
| DEFAULT_SCHEMA    | VARCHAR(128)  |                      | Nombre de esquema por omisión (QUALIFIER) utilizado para nombres no calificados en sentencias de SQL estáticas.                                                                                                                                             |
| VALID             | CHAR(1)       |                      | Y = Válido<br>N = No válido<br>X = El paquete no es operativo porque alguna instancia de función de la que depende se ha desactivado. Se necesita volver a ejecutar bind. Consulte la Nota 1 on "SYSCAT.PACKAGEDEP" en la página 1338                       |
| UNIQUE_ID         | CHAR(8)       |                      | Información de fecha y hora interna que indica cuándo se ha creado el paquete por primera vez.                                                                                                                                                              |
| TOTAL_SECT        | SMALLINT      |                      | Número total de secciones en el paquete.                                                                                                                                                                                                                    |
| FORMAT            | CHAR(1)       |                      | Formato de fecha y hora asociado con el paquete:<br>0 = Formato asociado con el código del país de la base de datos<br>1 = Hora y fecha USA<br>2 = Fecha EUR, hora EUR<br>3 = Fecha ISO, hora ISO<br>4 = Fecha JIS, hora JIS<br>5 = Fecha LOCAL, hora LOCAL |

## SYSCAT.PACKAGES

Tabla 77. Vista de catálogo SYSCAT.PACKAGES (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                   |
|-------------------|---------------|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ISOLATION         | CHAR(2)       | Sí                   | Nivel de aislamiento:<br>RR = Lectura repetible<br>RS = Estabilidad de lectura<br>CS = Estabilidad del cursor<br>UR = Lectura no confirmada                                                   |
| BLOCKING          | CHAR(1)       | Sí                   | Opción de bloqueo del cursor:<br>N = Sin bloqueo<br>U = Bloqueo de cursores no ambiguos<br>B = Bloqueo de todos los cursores                                                                  |
| INSERT_BUF        | CHAR(1)       |                      | Opción de inserción utilizada durante el enlace lógico:<br>Y = Las inserciones se ponen en almacenamiento intermedio<br>N = Las inserciones no se ponen en almacenamiento intermedio          |
| LANG_LEVEL        | CHAR(1)       | Sí                   | Valor LANGLEVEL utilizado durante BIND:<br>0 = SAA1<br>1 = SQL92E o MIA                                                                                                                       |
| FUNC_PATH         | VARCHAR(254)  |                      | La vía de acceso de SQL utilizada por el último mandato BIND para este paquete. Se utiliza como la vía de acceso por omisión para REBIND. SYSIBM para los paquetes anteriores a la Versión 2. |
| QUERYOPT          | INTEGER       |                      | La clase de optimización bajo la que se ha enlazado este paquete. Utilizada para volver a enlazar. Las clases son: 0, 1, 3, 5 y 9.                                                            |
| EXPLAIN_LEVEL     | CHAR(1)       |                      | Indica si se ha invocado Explain utilizando la opción de enlace EXPLAIN o EXPLSNAP.<br>P = Nivel de selección de planificación<br>En blanco si 'No' se invoca Explain                         |
| EXPLAIN_MODE      | CHAR(1)       |                      | Valor de la opción de enlace EXPLAIN:<br>Y = Sí (estática)<br>N = No<br>A = Todos (estática y dinámica)                                                                                       |

Tabla 77. Vista de catálogo SYSCAT.PACKAGES (continuación)

| Nombre de columna  | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                                         |
|--------------------|---------------|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EXPLAIN_SNAPSHOT   | CHAR(1)       |                      | Valor de la opción de enlace EXPLSNAP:<br>Y = Sí (estática)<br>N = No<br>A = Todos (estática y dinámica)                                                                                                                                                                            |
| SQLWARN            | CHAR(1)       |                      | ¿Se devuelven a la aplicación los SQLCODE positivos resultantes de las sentencias de SQL dinámicas?<br>Y = Sí<br>N = No, se suprimen                                                                                                                                                |
| SQLMATHWARN        | CHAR(1)       |                      | Valor del parámetro de configuración de la base de datos, DFT_SQLMATHWARN, en tiempo de enlace lógico. ¿Se tratan los errores aritméticos y los errores de conversión de recuperación de las sentencias de SQL estáticas como nulos con un aviso?<br>Y = Sí<br>N = No, se suprimen  |
| EXPLICIT_BIND_TIME | TIMESTAMP     |                      | La hora en la que este paquete se ha enlazado o vuelto a enlazar explícitamente por última vez. Cuando el paquete se vuelva a enlazar explícitamente, no se seleccionará ninguna instancia de función que se haya creado con posterioridad a esta hora.                             |
| LAST_BIND_TIME     | TIMESTAMP     |                      | La hora en la que el paquete se ha enlazado o vuelto a enlazar explícita o implícitamente por última vez.                                                                                                                                                                           |
| CODEPAGE           | SMALLINT      |                      | Página de códigos de la aplicación en tiempo de enlace (-1 si no se conoce).                                                                                                                                                                                                        |
| DEGREE             | CHAR(5)       |                      | Indica el límite en el paralelismo intrapartición (como opción de enlace lógico) cuando se ha enlazado el paquete.<br>1 = Sin paralelismo intrapartición.<br>2 - 32 767 = Grado de paralelismo intrapartición.<br>ANY = El grado se ha determinado por el gestor de bases de datos. |

## SYSCAT.PACKAGES

Tabla 77. Vista de catálogo SYSCAT.PACKAGES (continuación)

| Nombre de columna | Tipo de datos  | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------------------|----------------|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MULTINODE_PLANS   | CHAR(1)        |                      | Y = El paquete se ha vinculado en un entorno de múltiples particiones.<br>N = El paquete se ha vinculado en un entorno de una sola partición.                                                                                                                                                                                                                                                                                                                                                                                  |
| INTRA_PARALLEL    | CHAR(1)        |                      | Indica la utilización del paralelismo intrapartición por las sentencias de SQL estáticas dentro del paquete.<br>Y = una o varias sentencias de SQL estáticas del paquete utilizan paralelismo intrapartición.<br>N = ninguna sentencia de SQL estática del paquete utiliza paralelismo intrapartición.<br>F = una o varias sentencias de SQL estáticas del paquete utilizan paralelismo intrapartición; se ha inhabilitado este paralelismo para su uso en un sistema que no está configurado para paralelismo intrapartición. |
| VALIDATE          | CHAR(1)        |                      | B = Toda comprobación debe realizarse durante BIND<br>R = Reservado                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| DYNAMICRULES      | CHAR(1)        |                      | B = Sentencias de SQL dinámicas se manejan como sentencias de SQL estáticas en tiempo de ejecución; se utiliza authid del enlazador.<br>R = Sentencias de SQL dinámicas se manejan como sentencias de SQL dinámicas en tiempo de ejecución; se utiliza authid del usuario que ejecuta la operación.<br><br>El valor inicial es R.                                                                                                                                                                                              |
| SQLERROR          | CHAR(1)        |                      | Indica la opción SQLERROR en el submandato más reciente que ha enlazado o reenlazado el paquete.<br>C = Reservado<br>N = Sin paquete                                                                                                                                                                                                                                                                                                                                                                                           |
| REFRESHAGE        | DECIMAL (20.6) |                      | Duración de la indicación de la hora indicando la longitud máxima del tiempo entre cuando una sentencia REFRESH TABLE se ejecuta para una tabla de resumen y cuando la tabla de resumen se utiliza en lugar de una tabla base.                                                                                                                                                                                                                                                                                                 |

Tabla 77. Vista de catálogo SYSCAT.PACKAGES (continuación)

| Nombre de columna | Tipo de datos | Posibi-<br>lidad de<br>nulos | Descripción                                    |
|-------------------|---------------|------------------------------|------------------------------------------------|
| REMARKS           | VARCHAR(254)  | Sí                           | Comentario suministrado por el usuario o nulo. |

## SYSCAT.PARTITIONMAPS

---

### SYSCAT.PARTITIONMAPS

Contiene una fila para cada mapa de particionamiento que se utiliza para distribuir las filas de las tablas entre las particiones de un grupo de nodos, basándose en la clave de particionamiento de tablas.

Tabla 78. Vista de catálogo SYSCAT.PARTITIONMAPS

| Nombre de columna | Tipo de datos                | Posibilidad de nulos | Descripción                                                                                                                                                                                                                               |
|-------------------|------------------------------|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PMAP_ID           | SMALLINT                     |                      | Identificador del mapa de particionamiento.                                                                                                                                                                                               |
| PARTITIONMAP      | LONG VARCHAR<br>FOR BIT DATA |                      | El mapa de particionamiento real, un vector de 4.096 enteros de dos bytes para un grupo de nodos de varios nodos. Para un grupo de nodos de un solo nodo, hay una entrada que indica el número de partición (o nodo) del nodo individual. |

**SYSCAT.PASSTHROUGH**

Esta vista de catálogo contiene información acerca de las autorizaciones para consultar fuentes de datos en sesiones de paso a través. Una restricción en la tabla base necesita que los valores de SERVER se correspondan con los valores de la columna SERVER de SYSCAT.SERVERS. Ningún campo de SYSCAT.PASSTHROUGH puede ser nulo.

Tabla 79. Columnas de la vista de catálogo SYSCAT.PASSTHROUGH

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                    |
|-------------------|---------------|----------------------|--------------------------------------------------------------------------------------------------------------------------------|
| GRANTOR           | VARCHAR(128)  |                      | ID de autorización del usuario que ha otorgado el privilegio.                                                                  |
| GRANTEE           | VARCHAR(128)  |                      | ID de autorización del usuario o del grupo que tiene el privilegio.                                                            |
| GRANTEETYPE       | CHAR(1)       |                      | Una letra que especifica el tipo al que se ha otorgado:<br>U = Se otorga a un usuario individual.<br>G = Se otorga a un grupo. |
| SERVERNAME        | VARCHAR(128)  |                      | Nombre de la fuente de datos para la que se está otorgando la autorización al usuario o al grupo.                              |

## SYSCAT.PROCEDURES

---

### SYSCAT.PROCEDURES

Contiene una fila para cada procedimiento almacenado que se ha creado.

Tabla 80. Vista de catálogo SYSCAT.PROCEDURES

| Nombre de columna | Tipo de datos                | Posibi-<br>lidad de<br>nulos | Descripción                                                                                                                     |
|-------------------|------------------------------|------------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| PROCSHEMA         | VARCHAR(128)                 |                              | Nombre de procedimiento calificado.                                                                                             |
| PROCNAME          | VARCHAR(128)                 |                              |                                                                                                                                 |
| SPECIFICNAME      | VARCHAR(18)                  |                              | Nombre de instancia del procedimiento (puede ser generado por el sistema).                                                      |
| PROCEDURE_ID      | INTEGER                      |                              | ID interno del procedimiento almacenado.                                                                                        |
| DEFINER           | VARCHAR(128)                 |                              | Autorización del encargado de definir el procedimiento.                                                                         |
| PARAM_COUNT       | SMALLINT                     |                              | Número de parámetros del procedimiento.                                                                                         |
| PARAM_SIGNATURE   | VARCHAR(180)<br>FOR BIT DATA |                              | Concatenación de un máximo de 90 tipos de parámetros, en formato interno. Longitud cero si el procedimiento no toma parámetros. |
| ORIGIN            | CHAR(1)                      |                              | Siempre 'E' = Definido por el usuario, externo                                                                                  |
| CREATE_TIME       | TIMESTAMP                    |                              | Indicación de la hora del registro del procedimiento.                                                                           |
| DETERMINISTIC     | CHAR(1)                      |                              | Y = Los resultados son determinantes.<br>N = Los resultados no son determinantes.                                               |
| FENCED            | CHAR(1)                      |                              | Y = Limitado<br>N = No limitado                                                                                                 |
| NULLCALL          | CHAR(1)                      |                              | Siempre Y = NULLCALL                                                                                                            |
| LANGUAGE          | CHAR(8)                      |                              | Implantación del cuerpo del lenguaje del procedimiento. Los valores posibles son:<br>C<br>COBOL<br>JAVA<br>SQL                  |
| IMPLEMENTATION    | VARCHAR(254)                 | Sí                           | Identifica la vía/módulo/función (LANGUAGE = C o COBOL) o método (LANGUAGE = JAVA) que implementa el procedimiento.             |
| CLASS             | VARCHAR(128)                 | Sí                           | Si LANGUAGE = JAVA entonces identifica la clase que implementa este procedimiento. En otro caso, es nulo.                       |



Tabla 80. Vista de catálogo SYSCAT.PROCEDURES (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                                                                              |
|-------------------|---------------|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| JAR_ID            | VARCHAR(128)  | Sí                   | Si LANGUAGE = JAVA entonces identifica el archivo jar que implementa este procedimiento. En otro caso, es nulo.                                                                                                                                                                                                          |
| PARAM_STYLE       | CHAR(8)       |                      | DB2DARI = El lenguaje es C<br>DB2GENRL = El lenguaje es Java<br>DB2SQL = El lenguaje es C o COBOL<br>JAVA = El lenguaje es Java o SQL<br>GENERAL = El lenguaje es C o COBOL<br>GNLRNULL = El lenguaje es C o COBOL                                                                                                       |
| CONTAINS_SQL      | CHAR(1)       |                      | Indica si un procedimiento contiene SQL.<br>C = CONTAINS SQL: sólo se permite SQL que no lee ni modifica datos SQL.<br>M = MODIFY SQL DATA: se permite todo el SQL permitido en procedimientos<br>N = NO SQL: no se permite SQL<br>R = READS SQL DATA: sólo se permite SQL que lee datos SQL                             |
| DBINFO            | CHAR(1)       |                      | Indica si se pasa un parámetro DBINFO al procedimiento.<br>N = DBINFO no se pasa<br>Y = DBINFO se pasa                                                                                                                                                                                                                   |
| PROGRAM_TYPE      | CHAR(1)       |                      | Indica cómo se invoca el procedimiento.<br>M = Principal<br>S = Subrutina                                                                                                                                                                                                                                                |
| RESULT_SETS       | SMALLINT      |                      | El límite superior estimado de los conjuntos del resultado devueltos.                                                                                                                                                                                                                                                    |
| VALID             | CHAR(1)       |                      | blank = no es un procedimiento SQL<br>Y = el procedimiento SQL es válido<br>N = el procedimiento SQL no es válido<br>X = El procedimiento SQL no es operativo porque se ha eliminado alguna instancia de función necesaria para el procedimiento. Se debe eliminar y volver a crear explícitamente el procedimiento SQL. |

## SYSCAT.PROCEDURES

Tabla 80. Vista de catálogo SYSCAT.PROCEDURES (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                      |
|-------------------|---------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TEXT_BODY_OFFSET  | INTEGER       |                      | Si es un procedimiento SQL, esta columna contiene el desplazamiento respecto al inicio del cuerpo del procedimiento SQL de la sentencia CREATE PROCEDURE. Si es un procedimiento externo, el valor es 0.         |
| TEXT              | CLOB(1M)      | Sí                   | Si es un procedimiento SQL, esta columna contiene el texto completo de la sentencia CREATE PROCEDURE, tal como está escrita. Es nula si el texto completo es más largo que 1M, o si es un procedimiento externo. |
| REMARKS           | VARCHAR(254)  | Sí                   | Comentario suministrado por el usuario o nulo.                                                                                                                                                                   |

**SYSCAT.PROCOPTIONS**

Cada fila contiene los valores de las opciones específicas de servidor

*Tabla 81. Vista de catálogo SYSCAT.PROCOPTIONS*

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                      |
|-------------------|---------------|----------------------|------------------------------------------------------------------|
| PROCSHEMA         | VARCHAR(128)  |                      | Calificador para el nombre o apodo del procedimiento almacenado. |
| PROCNAME          | VARCHAR(128)  |                      | Nombre o apodo del procedimiento almacenado.                     |
| OPTION            | VARCHAR(128)  |                      | Nombre de la opción de procedimiento almacenado.                 |
| SETTING           | VARCHAR(255)  |                      | Valor de la opción de procedimiento almacenado.                  |

## SYSCAT.PROCPARMOPTIONS

---

### SYSCAT.PROCPARMOPTIONS

Cada fila contiene los valores de las opciones específicas del parámetro de procedimiento.

*Tabla 82. Vista de catálogo SYSCAT.PROCPARMOPTIONS*

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                              |
|-------------------|---------------|----------------------|--------------------------------------------------------------------------|
| PROCSHEMA         | VARCHAR(128)  |                      | Nombre o apodo de procedimiento calificado.                              |
| PROCNAME          | VARCHAR(128)  |                      |                                                                          |
| ORDINAL           | SMALLINT      |                      | Posición numérica del parámetro dentro de la signaturadel procedimiento. |
| OPTION            | VARCHAR(128)  |                      | Nombre de la opción de procedimiento almacenado.                         |
| SETTING           | VARCHAR(255)  |                      | Valor.                                                                   |

## SYSCAT.PROCPARMS

Contiene una fila para cada parámetro de un procedimiento almacenado.

Tabla 83. Vista de catálogo SYSCAT.PROCPARMS

| Nombre de columna | Tipo de datos | Posibi-<br>lidad de<br>nulos | Descripción                                                                                                                                       |
|-------------------|---------------|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| PROCSHEMA         | VARCHAR(128)  |                              | Nombre de procedimiento calificado.                                                                                                               |
| PROCNAME          | VARCHAR(128)  |                              |                                                                                                                                                   |
| SPECIFICNAME      | VARCHAR(18)   |                              | Nombre de instancia del procedimiento (puede ser generado por el sistema).                                                                        |
| SERVERNAME        | VARCHAR(128)  | Sí                           | Nombre de la fuente de datos en la que reside el procedimiento almacenado.                                                                        |
| ORDINAL           | SMALLINT      |                              | Posición numérica del parámetro dentro de la signatura del método.                                                                                |
| PARAMNAME         | VARCHAR(18)   |                              | Nombre de parámetro.                                                                                                                              |
| TYPESHEMA         | VARCHAR(128)  |                              | Nombre calificado del tipo de datos del parámetro.                                                                                                |
| TYPENAME          | VARCHAR(18)   |                              |                                                                                                                                                   |
| TYPEID            | SMALLINT      | Sí                           | ID de tipo interno.                                                                                                                               |
| SOURCETYPEID      | SMALLINT      | Sí                           | ID de tipo interno del tipo fuente. Nulo para tipos incorporados.                                                                                 |
| NULLS             | CHAR(1)       |                              | Base de datos federada regla con posibilidad de contener nulos:<br>Y = Con posibilidad de contener nulos<br>N = Sin posibilidad de contener nulos |
| LENGTH            | INTEGER       |                              | Longitud del parámetro.                                                                                                                           |
| SCALE             | SMALLINT      |                              | Escala del parámetro.                                                                                                                             |
| PARAM_MODE        | VARCHAR(5)    |                              | IN = Entrada<br>OUT = Salida<br>INOUT = Entrada/salida                                                                                            |
| CODEPAGE          | SMALLINT      |                              | Página de códigos del parámetro. 0 indica no aplicable o un parámetro para los datos de caracteres declarados con el atributo FOR BIT DATA.       |
| DBCS_CODEPAGE     | SMALLINT      | Sí                           | Página de códigos DBCS. Nulo para campos numéricos.                                                                                               |
| AS_LOCATOR        | CHAR(1)       |                              | Siempre 'N'                                                                                                                                       |

## SYSCAT.PROCPARMS

Tabla 83. Vista de catálogo SYSCAT.PROCPARMS (continuación)

| Nombre de columna | Tipo de datos | Posibi-<br>lidad de<br>nulos | Descripción                                                                                                                         |
|-------------------|---------------|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| TARGET_TYPESHEMA  | VARCHAR(128)  | Sí                           | Si el tipo de parámetro es de referencia, entonces contiene un nombre calificado de tipo de fila de destino. En otro caso, es nulo. |
| TARGET_TYPENAME   | VARCHAR(18)   |                              |                                                                                                                                     |
| SCOPE_TABSCHEMA   | VARCHAR(128)  | Sí                           | Si el tipo de parámetro es de referencia, entonces contiene nombre calificado ámbito (tabla de destino). En otro caso, es nulo.     |
| SCOPE_TABNAME     | VARCHAR(128)  |                              |                                                                                                                                     |

## SYSCAT.REFERENCES

Contiene una fila para cada restricción de referencia definida.

Tabla 84. Vista de catálogo SYSCAT.REFERENCES

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                          |
|-------------------|---------------|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CONSTNAME         | VARCHAR(18)   |                      | Nombre de restricción.                                                                                                                                                               |
| TABSHEMA          | VARCHAR(128)  |                      | Nombre calificado de la restricción.                                                                                                                                                 |
| TABNAME           | VARCHAR(128)  |                      |                                                                                                                                                                                      |
| DEFINER           | VARCHAR(128)  |                      | Usuario que ha creado la restricción.                                                                                                                                                |
| REFKEYNAME        | VARCHAR(18)   |                      | Nombre de la clave padre.                                                                                                                                                            |
| REFTABSHEMA       | VARCHAR(128)  |                      | Nombre de la tabla padre.                                                                                                                                                            |
| REFTABNAME        | VARCHAR(128)  |                      |                                                                                                                                                                                      |
| COLCOUNT          | SMALLINT      |                      | Número de columnas de la clave foránea.                                                                                                                                              |
| DELETERULE        | CHAR(1)       |                      | Regla de supresión.<br>A = NO ACTION<br>C = CASCADE<br>N = SET NULL<br>R = RESTRICT                                                                                                  |
| UPDATERULE        | CHAR(1)       |                      | Regla de actualización:<br>A = NO ACTION<br>R = RESTRICT                                                                                                                             |
| CREATE_TIME       | TIMESTAMP     |                      | La indicación de la hora cuando se ha definido la restricción de referencia.                                                                                                         |
| FK_COLNAMES       | VARCHAR(640)  |                      | Lista de nombres de columna de clave foránea. Aviso: Esta columna se eliminará en el futuro. Utilice el apartado "SYSCAT.KEYCOLUSE" en la página 1333 para ampliar esta información. |
| PK_COLNAMES       | VARCHAR(640)  |                      | Lista de nombres de columna de clave padre. Aviso: Esta columna se eliminará en el futuro. Utilice el apartado "SYSCAT.KEYCOLUSE" en la página 1333 para ampliar esta información.   |

**Nota:**

1. La vista SYSCAT.REFERENCES se basa en la tabla SYSIBM.SYSRELS de la Versión 1.

## SYSCAT.REVTYPEMAPPINGS

### SYSCAT.REVTYPEMAPPINGS

Cada fila contiene las correlaciones de tipos de datos invertidos (correlaciones de tipos de datos definidos localmente para los tipos de datos de fuente de datos). No hay datos en esta versión. Se define para una posible utilización futura con las correlaciones de tipos de datos.

Tabla 85. Vista de catálogo SYSCAT.REVTYPEMAPPINGS

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                              |
|-------------------|---------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TYPE_MAPPING      | VARCHAR(18)   |                      | Nombre de la correlación de tipos invertidos (puede ser generado por el sistema).                                                                                                        |
| TYPESCHEMA        | VARCHAR(128)  | Sí                   | Nombre de esquema del tipo. Nulo para tipos incorporados del sistema.                                                                                                                    |
| TYPENAME          | VARCHAR(18)   |                      | Nombre del tipo local de una correlación de tipos invertidos.                                                                                                                            |
| TYPEID            | SMALLINT      |                      | Identificador de tipo.                                                                                                                                                                   |
| SOURCETYPEID      | SMALLINT      |                      | Identificador de tipo de fuente.                                                                                                                                                         |
| DEFINER           | VARCHAR(128)  |                      | ID de autorización bajo el cual se ha creado esta correlación de tipos.                                                                                                                  |
| LOWER_LEN         | INTEGER       | Sí                   | Límite inferior de la longitud/precisión del tipo local.                                                                                                                                 |
| UPPER_LEN         | INTEGER       | Sí                   | Límite superior de la longitud/precisión del tipo local. Si es nulo, el sistema determina el mejor atributo de longitud/precisión.                                                       |
| LOWER_SCALE       | SMALLINT      | Sí                   | Límite inferior de la escala para los tipos de datos decimales locales.                                                                                                                  |
| UPPER_SCALE       | SMALLINT      | Sí                   | Límite superior de la escala para tipos de datos decimales locales. Si es nulo, el sistema determina el mejor atributo de escala.                                                        |
| S_OPR_P           | CHAR(2)       | Sí                   | Relación entre la escala local y la precisión local. Se pueden utilizar operadores de comparación básicos. Un nulo indica que no es necesaria ninguna relación específica.               |
| BIT_DATA          | CHAR(1)       | Sí                   | Y = El tipo es para datos de bits.<br>N = El tipo no es para datos de bits.<br>NULL = No se trata de un tipo de datos de caracteres o el sistema determina el atributo de datos de bits. |
| WRAPNAME          | VARCHAR(128)  | Sí                   | La correlación se aplica a este protocolo de acceso de datos.                                                                                                                            |



Tabla 85. Vista de catálogo SYSCAT.REVTYPEMAPPINGS (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                              |
|-------------------|---------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SERVERNAME        | VARCHAR(128)  | Sí                   | Nombre de la fuente de datos.                                                                                                                                                            |
| SERVERTYPE        | VARCHAR (30)  | Sí                   | La correlación se aplica a este tipo de fuente de datos.                                                                                                                                 |
| SERVERVERSION     | VARCHAR(18)   | Sí                   | La correlación se aplica a esta versión de SERVERTYPE.                                                                                                                                   |
| REMOTE_TYPESHEMA  | VARCHAR(128)  | Sí                   | Nombre de esquema del tipo remoto.                                                                                                                                                       |
| REMOTE_TYPENAME   | VARCHAR(128)  |                      | Nombre del tipo de datos tal como está definido en la fuente o fuentes de datos.                                                                                                         |
| REMOTE_META_TYPE  | CHAR(1)       | Sí                   | S = El tipo remoto es un tipo incorporado del sistema.<br>T = El tipo remoto es un tipo diferenciado.                                                                                    |
| REMOTE_LENGTH     | INTEGER       | Sí                   | Número máximo de dígitos para el tipo decimal remoto y el número máximo de caracteres para el tipo de caracteres remoto. De lo contrario es nulo.                                        |
| REMOTE_SCALE      | SMALLINT      | Sí                   | Número máximo de dígitos permitidos a la derecha de la coma decimal (para tipos decimales remotos). De lo contrario es nulo.                                                             |
| REMOTE_BIT_DATA   | CHAR(1)       | Sí                   | Y = El tipo es para datos de bits.<br>N = El tipo no es para datos de bits.<br>NULL = No se trata de un tipo de datos de caracteres o el sistema determina el atributo de datos de bits. |
| USER_DEFINED      | CHAR(1)       |                      | Definido por el usuario.                                                                                                                                                                 |
| CREATE_TIME       | TIMESTAMP     |                      | Hora en que se ha creado la correlación.                                                                                                                                                 |
| REMARKS           | VARCHAR(254)  | Sí                   | Comentario suministrado por el usuario o nulo.                                                                                                                                           |

## SYSCAT.SCHEMAAUTH

---

### SYSCAT.SCHEMAAUTH

Contiene una o varias filas para cada usuario o grupo a los que se ha otorgado un privilegio para un esquema determinado de la base de datos. Todos los privilegios para un esquema individual otorgados por un otorgante específico a un receptor del otorgamiento específico aparecen en una sola fila.

Tabla 86. Vista de catálogo SYSCAT.SCHEMAAUTH

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                              |
|-------------------|---------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GRANTOR           | VARCHAR(128)  |                      | ID de autorización del usuario que ha otorgado los privilegios o SYSIBM.                                                                                                                 |
| GRANTEE           | VARCHAR(128)  |                      | ID de autorización del usuario o grupo que tiene los privilegios.                                                                                                                        |
| GRANTEETYPE       | CHAR(1)       |                      | U = Se otorga a un usuario individual.<br>G = Se otorga a un grupo.                                                                                                                      |
| SCHEMANAME        | VARCHAR(128)  |                      | Nombre del esquema.                                                                                                                                                                      |
| ALTERINAUTH       | CHAR(1)       |                      | Indica si receptor del otorgamiento tiene el privilegio ALTERIN en el esquema:<br>Y = Tiene el privilegio.<br>G = Tiene el privilegio y es otorgable.<br>N = No tiene el privilegio.     |
| CREATEINAUTH      | CHAR(1)       |                      | Indica si el receptor del otorgamiento tiene el privilegio CREATEIN en el esquema:<br>Y = Tiene el privilegio.<br>G = Tiene el privilegio y es otorgable.<br>N = No tiene el privilegio. |
| DROPINAUTH        | CHAR(1)       |                      | Indica si el receptor del otorgamiento tiene el privilegio DROPIN en el esquema:<br>Y = Tiene el privilegio.<br>G = Tiene el privilegio y es otorgable.<br>N = No tiene el privilegio.   |

---

**SYSCAT.SCHEMATA**

Contiene una fila para cada esquema.

*Tabla 87. Vista de catálogo SYSCAT.SCHEMATA*

| Nombre de columna | Tipo de datos | Posibi-<br>lidad de<br>nulos | Descripción                                                                                  |
|-------------------|---------------|------------------------------|----------------------------------------------------------------------------------------------|
| SCHEMANAME        | VARCHAR(128)  |                              | Nombre del esquema.                                                                          |
| OWNER             | VARCHAR(128)  |                              | ID de autorización del esquema. El valor para los esquemas creados implícitamente es SYSIBM. |
| DEFINER           | VARCHAR(128)  |                              | Usuario que ha creado el esquema.                                                            |
| CREATE_TIME       | TIMESTAMP     |                              | Indicación de la hora en la que se ha creado el objeto.                                      |
| REMARKS           | VARCHAR(254)  | Sí                           | Comentario proporcionado por el usuario.                                                     |

## SYSCAT.SEQUENCES

---

### SYSCAT.SEQUENCES

La vista SYSCAT.SEQUENCES se genera automáticamente para las bases de datos creadas con el FixPak 3 o posterior.

Para las bases de datos creadas antes del FixPak 3, ejecute el mandato db2updv7 para añadir la vista a la base de datos. Consulte las actualizaciones de la publicación *Consulta de mandatos* en las Notas del release para obtener detalles.

Esta vista de catálogo se actualiza durante las operaciones normales, en respuesta a las sentencias de definición de datos de SQL, a las rutinas de entorno y a determinados programas de utilidad. Los datos de la vista de catálogo están disponible mediante los recursos normales de consulta de SQL. Las columnas tienen nombres coherentes basados en el tipo de objetos que describen.

Tabla 88. Columnas de la vista de catálogo SYSCAT.SEQUENCES

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                         |
|-------------------|---------------|----------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| SEQSHEMA          | VARCHAR(128)  |                      | Esquema de la secuencia.                                                                                                            |
| SEQNAME           | VARCHAR(128)  |                      | Nombre de secuencia (generado por DB2 para una columna de identificación).                                                          |
| DEFINER           | VARCHAR(128)  |                      | Definidor de la secuencia.                                                                                                          |
| OWNER             | VARCHAR(128)  |                      | Propietario de la secuencia.                                                                                                        |
| SEQID             | INTEGER       |                      | ID interno de la secuencia.                                                                                                         |
| SEQTYPE           | CHAR(1)       |                      | Tipo de secuencia<br>S = Secuencia corriente                                                                                        |
| INCREMENT         | DECIMAL(31,0) |                      | Valor de incremento.                                                                                                                |
| START             | DECIMAL(31,0) |                      | Valor inicial.                                                                                                                      |
| MAXVALUE          | DECIMAL(31,0) | Sí                   | Valor máximo.                                                                                                                       |
| MINVALUE          | DECIMAL(31,0) |                      | Valor mínimo.                                                                                                                       |
| CYCLE             | CHAR(1)       |                      | Indica si se producirá el ciclo cuando se alcance un límite:<br>Y - se producirá el ciclo<br>N - no se producirá el ciclo           |
| CACHE             | INTEGER       |                      | Número de valores de secuencia que se deben preasignar en memoria para el acceso más rápido. 0 indica que no se preasignan valores. |

Tabla 88. Columnas de la vista de catálogo SYSCAT.SEQUENCES (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                         |
|-------------------|---------------|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ORDER             | CHAR(1)       |                      | Indica si los números de secuencia deben generarse según el orden de petición:<br>Y - los números de secuencia deben generarse según el orden de petición<br>N - no es necesario que los números de secuencia se generen según el orden de petición |
| DATATYPEID        | INTEGER       |                      | Para tipos incorporados, ID interno del tipo incorporado. Para tipos diferenciados, ID interno del tipo diferenciado.                                                                                                                               |
| SOURCETYPEID      | INTEGER       |                      | Para un tipo incorporado, tiene un valor de 0. Para un tipo diferenciado, es el ID interno del tipo incorporado que es el tipo fuente para el tipo diferenciado.                                                                                    |
| CREATE_TIME       | TIMESTAMP     |                      | Hora en que se ha creado la secuencia.                                                                                                                                                                                                              |
| ALTER_TIME        | TIMESTAMP     |                      | Hora en que se ha ejecutado la última sentencia ALTER SEQUENCE para esta secuencia.                                                                                                                                                                 |
| PRECISION         | SMALLINT      |                      | Precisión del tipo de datos de la secuencia. Los valores son: 5 para SMALLINT, 10 para INTEGER y 19 para BIGINT. Para DECIMAL, es la precisión del tipo de datos DECIMAL especificado.                                                              |
| ORIGIN            | CHAR(1)       |                      | Origen de la secuencia<br>U - Secuencia generada por el usuario<br>S - Secuencia generada por el sistema                                                                                                                                            |
| REMARKS           | VARCHAR(254)  | Sí                   | Comentarios proporcionados por el usuario o nulo.                                                                                                                                                                                                   |

## SYSCAT.SERVEROPTIONS

---

### SYSCAT.SERVEROPTIONS

Cada fila contiene las opciones de configuración a nivel de servidor.

Tabla 89. Columnas de la vista de catálogo SYSCAT.SERVEROPTIONS

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                    |
|-------------------|---------------|----------------------|------------------------------------------------|
| WRAPNAME          | VARCHAR(128)  | Sí                   | Nombre de reiniciador.                         |
| SERVERNAME        | VARCHAR(128)  | Sí                   | Nombre del servidor.                           |
| SERVERTYPE        | VARCHAR (30)  | Sí                   | Tipo de servidor.                              |
| SERVERVERSION     | VARCHAR(18)   | Sí                   | Versión de servidor.                           |
| CREATE_TIME       | TIMESTAMP     |                      | Hora en que se ha creado la entrada.           |
| OPTION            | VARCHAR(128)  |                      | Nombre de la opción de servidor.               |
| SETTING           | VARCHAR(2048) |                      | Valor de la opción de servidor.                |
| SERVEROPTIONKEY   | VARCHAR(18)   |                      | Identifica exclusivamente una fila.            |
| REMARKS           | VARCHAR(254)  | Sí                   | Comentario suministrado por el usuario o nulo. |

**SYSCAT.SERVERS**

Cada fila representa una fuente de datos. Las entradas del catálogo no son necesarias para las tablas que se almacenan en la misma instancia que contiene esta tabla del catálogo.

Tabla 90. Columnas de la vista de catálogo SYSCAT.SERVERS

| Nombre        | Tipo de datos | Posibilidad de nulos | Descripción                                           |
|---------------|---------------|----------------------|-------------------------------------------------------|
| WRAPNAME      | VARCHAR(128)  |                      | Nombre de reiniciador.                                |
| SERVERNAME    | VARCHAR(128)  |                      | Nombre de la fuente de datos conocida por el sistema. |
| SERVERTYPE    | VARCHAR (30)  | Sí                   | Tipo de fuente de datos (siempre en mayúsculas).      |
| SERVERVERSION | VARCHAR(18)   | Sí                   | Versión de la fuente de datos.                        |
| REMARKS       | VARCHAR(254)  | Sí                   | Comentario suministrado por el usuario o nulo.        |

## SYSCAT.STATEMENTS

---

### SYSCAT.STATEMENTS

Contiene una o varias filas para cada sentencia de SQL de cada paquete de la base de datos.

Tabla 91. Vista de catálogo SYSCAT.STATEMENTS

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                          |
|-------------------|---------------|----------------------|--------------------------------------------------------------------------------------|
| PKGSHEMA          | VARCHAR(128)  |                      | Nombre del paquete.                                                                  |
| PKGNAME           | CHAR(8)       |                      |                                                                                      |
| STMTNO            | INTEGER       |                      | Número de línea de la sentencia de SQL del módulo fuente del programa de aplicación. |
| SECTNO            | SMALLINT      |                      | Número de la sección del paquete que contiene la sentencia de SQL.                   |
| SEQNO             | SMALLINT      |                      | Siempre 1.                                                                           |
| TEXT              | CLOB(64K)     |                      | Texto de la sentencia de SQL.                                                        |



## SYSCAT.TABAUTH

Contiene una o varias filas para cada usuario o grupo a los que se otorga un privilegio para una tabla o vista determinada de la base de datos. Todos los privilegios para una tabla o vista individual otorgados por un otorgante específico a un usuario autorizado específico aparecen en una sola fila.

Tabla 92. Vista de catálogo SYSCAT.TABAUTH

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                  |
|-------------------|---------------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GRANTOR           | VARCHAR(128)  |                      | ID de autorización del usuario que ha otorgado los privilegios o SYSIBM.                                                                                                                     |
| GRANTEE           | VARCHAR(128)  |                      | ID de autorización del usuario o grupo que tiene los privilegios.                                                                                                                            |
| GRANTEETYPE       | CHAR(1)       |                      | U = Se otorga a un usuario individual.<br>G = Se otorga a un grupo.                                                                                                                          |
| TABSCHEMA         | VARCHAR(128)  |                      | Nombre calificado de la tabla o vista.                                                                                                                                                       |
| TABNAME           | VARCHAR(128)  |                      |                                                                                                                                                                                              |
| CONTROLAUTH       | CHAR(1)       |                      | Indica si el receptor del otorgamiento tiene el privilegio CONTROL en la tabla o vista:<br>Y = Tiene el privilegio.<br>N = No tiene el privilegio.                                           |
| ALTERAUTH         | CHAR(1)       |                      | Indica si el receptor del otorgamiento tiene el privilegio ALTER en la tabla:<br>Y = Tiene el privilegio.<br>N = No tiene el privilegio.<br>G = Tiene el privilegio y es otorgable.          |
| DELETEAUTH        | CHAR(1)       |                      | Indica si el receptor del otorgamiento tiene el privilegio DELETE en la tabla o vista:<br>Y = Tiene el privilegio.<br>N = No tiene el privilegio.<br>G = Tiene el privilegio y es otorgable. |
| INDEXAUTH         | CHAR(1)       |                      | Indica si el receptor del otorgamiento tiene el privilegio INDEX en la tabla:<br>Y = Tiene el privilegio.<br>N = No tiene el privilegio.<br>G = Tiene el privilegio y es otorgable.          |

## SYSCAT.TABAUTH

Tabla 92. Vista de catálogo SYSCAT.TABAUTH (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                     |
|-------------------|---------------|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| INSERTAUTH        | CHAR(1)       |                      | Indica si el receptor del otorgamiento tiene el privilegio INSERT en la tabla o vista:<br>Y = Tiene el privilegio.<br>N = No tiene el privilegio.<br>G = Tiene el privilegio y es otorgable.    |
| SELECTAUTH        | CHAR(1)       |                      | Indica si el receptor del otorgamiento tiene el privilegio SELECT en la tabla o vista:<br>Y = Tiene el privilegio.<br>N = No tiene el privilegio.<br>G = Tiene el privilegio y es otorgable.    |
| REFAUTH           | CHAR(1)       |                      | Indica si el receptor del otorgamiento tiene el privilegio REFERENCE en la tabla o vista:<br>Y = Tiene el privilegio.<br>N = No tiene el privilegio.<br>G = Tiene el privilegio y es otorgable. |
| UPDATEAUTH        | CHAR(1)       |                      | Indica si el receptor del otorgamiento tiene el privilegio UPDATE en la tabla o vista:<br>Y = Tiene el privilegio.<br>N = No tiene el privilegio.<br>G = Tiene el privilegio y es otorgable.    |

**SYSCAT.TABCONST**

Cada fila representa una restricción de tabla del tipo CHECK, UNIQUE, PRIMARY KEY o FOREIGN KEY.

Tabla 93. Vista de catálogo SYSCAT.TABCONST

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                 |
|-------------------|---------------|----------------------|---------------------------------------------------------------------------------------------|
| CONSTNAME         | VARCHAR(18)   |                      | Nombre de la restricción (exclusivo en una tabla).                                          |
| TABSHEMA          | VARCHAR(128)  |                      | Nombre calificado de la tabla a la que se aplica esta restricción.                          |
| TABNAME           | VARCHAR(128)  |                      |                                                                                             |
| DEFINER           | VARCHAR(128)  |                      | ID de autorización bajo el cual se ha definido la restricción.                              |
| TYPE              | CHAR(1)       |                      | Indica el tipo de restricción:<br>F= FOREIGN KEY<br>K= CHECK<br>P= PRIMARY KEY<br>U= UNIQUE |
| REMARKS           | VARCHAR(254)  | Sí                   | Comentario suministrado por el usuario o nulo.                                              |

## SYSCAT.TABLES

---

### SYSCAT.TABLES

Contiene una fila para cada tabla, vista, apodo o seudónimo que se crea. Todas las tablas y vistas de catálogo tienen entradas en la vista de catálogo SYSCAT.TABLES.

Tabla 94. Vista de catálogo SYSCAT.TABLES

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                              |
|-------------------|---------------|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TABSCHEMA         | VARCHAR(128)  |                      | Nombre calificado de la tabla, vista, apodo o seudónimo.                                                                                                                 |
| TABNAME           | VARCHAR(128)  |                      |                                                                                                                                                                          |
| DEFINER           | VARCHAR(128)  |                      | Usuario que ha creado la tabla, vista, apodo o seudónimo.                                                                                                                |
| TYPE              | CHAR(1)       |                      | El tipo de objeto:<br>A = Seudónimo<br>H = Tabla de jerarquía<br>N = Apodo<br>S = Tabla de resumen<br>T = Tabla<br>U = Tabla con tipo<br>V = Vista<br>W = Vista con tipo |
| STATUS            | CHAR(1)       |                      | El tipo de objeto:<br>N = Tabla, vista, seudónimo o apodo normal<br>C = Pendiente de comprobación en tabla o apodo<br>X = Apodo o vista no operativa                     |
| BASE_TABSCHEMA    | VARCHAR(128)  | Sí                   | Si TYPE=A, estas columnas identifican la tabla, vista, seudónimo o apodo a los que hace referencia este seudónimo; de lo contrario son nulos.                            |
| BASE_TABNAME      | VARCHAR(128)  | Sí                   |                                                                                                                                                                          |
| ROWTYPESCHEMA     | VARCHAR(128)  | Sí                   | Contiene el nombre calificado del tipo de fila de esta tabla, donde sea aplicable. En otro caso, es nulo.                                                                |
| ROWTYPENAME       | VARCHAR(18)   |                      |                                                                                                                                                                          |
| CREATE_TIME       | TIMESTAMP     |                      | La indicación de la hora en la que se ha creado el objeto.                                                                                                               |
| STATS_TIME        | TIMESTAMP     | Sí                   | La última vez que se ha realizado un cambio en las estadísticas registradas para esta tabla. Nulo si no hay estadísticas disponibles.                                    |
| COLCOUNT          | SMALLINT      |                      | Número de columnas en la tabla.                                                                                                                                          |

Tabla 94. Vista de catálogo SYSCAT.TABLES (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                                        |
|-------------------|---------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TABLEID           | SMALLINT      |                      | Identificador interno de tabla.                                                                                                                                                                                                                                                    |
| TBSPACEID         | SMALLINT      |                      | Identificador interno de espacio de tablas principal para esta tabla.                                                                                                                                                                                                              |
| CARD              | BIGINT        |                      | Número total de filas en la tabla. Para tablas en una tabla de jerarquía, su número de filas del nivel determinado de la jerarquía; -1 si no se reúnen las estadísticas o la fila describe una vista o seudónimo; -2 para tablas de jerarquía (tablas-H)                           |
| NPAGES            | INTEGER       |                      | Número total de páginas en las que existen las filas de la tabla; -1 si no se reúnen estadísticas o la fila describe una vista o un seudónimo; -2 para subtablas o tablas-H.                                                                                                       |
| FPAGES            | INTEGER       |                      | Número total de páginas; -1 si no se reúnen estadísticas o la fila describe una vista o un seudónimo; -2 para subtablas o tablas-H.                                                                                                                                                |
| OVERFLOW          | INTEGER       |                      | Número total de registros de desbordamiento en la tabla; -1 si no se reúnen estadísticas o la fila describe una vista o un seudónimo; -2 para subtablas o tablas-H.                                                                                                                |
| TBSPACE           | VARCHAR(18)   | Sí                   | Nombre de espacio de tablas principal para la tabla. Si no se especifica ningún otro espacio de tablas, todas las partes de la tabla se almacenan en este espacio de tablas. Es nulo para seudónimos y vistas.                                                                     |
| INDEX_TBSPACE     | VARCHAR(18)   | Sí                   | Nombre del espacio de tablas que contiene todos los índices creados en esta tabla. Nulo para los seudónimos y vistas o si se ha omitido la cláusula INDEX IN o se ha especificado con el mismo valor que la cláusula IN de la sentencia CREATE TABLE.                              |
| LONG_TBSPACE      | VARCHAR(18)   | Sí                   | Nombre del espacio de tablas que contiene todos los datos largos (tipos de columna LONG o LOB) para esta tabla. Nulo para los seudónimos y vistas, o si se ha omitido la cláusula LONG IN o se ha especificado con el mismo valor que la cláusula IN de la sentencia CREATE TABLE. |
| PARENTS           | SMALLINT      | Sí                   | Número de tablas padre de esta tabla (el número de restricciones de referencia de las que depende esta tabla).                                                                                                                                                                     |

## SYSCAT.TABLES

Tabla 94. Vista de catálogo SYSCAT.TABLES (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-------------------|---------------|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CHILDREN          | SMALLINT      | Sí                   | Número de tablas dependientes de esta tabla (el número de restricciones de referencia en las que esta tabla es padre).                                                                                                                                                                                                                                                                                                                                                                                                                     |
| SELFREFS          | SMALLINT      | Sí                   | Número de restricciones de referencia propia para esta tabla (el número de restricciones de referencia en las que esta tabla es tanto padre como dependiente).                                                                                                                                                                                                                                                                                                                                                                             |
| KEYCOLUMNS        | SMALLINT      | Sí                   | Número de columnas de la clave primaria de la tabla.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| KEYINDEXID        | SMALLINT      | Sí                   | ID de índice del índice principal. Este campo es nulo o 0 si no hay clave primaria.                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| KEYUNIQUE         | SMALLINT      |                      | Número de restricciones de unicidad (distintas de la clave primaria) definidas en esta tabla.                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| CHECKCOUNT        | SMALLINT      |                      | Número de restricciones de comprobación definidas en esta tabla.                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| DATA_CAPTURE      | CHAR(1)       |                      | <p>Y = La tabla participa en la duplicación de datos</p> <p>N = No participa</p> <p>L = La tabla interviene en la duplicación de datos, incluida la duplicación de las columnas LONG VARCHAR y LONG VARCHARIC</p>                                                                                                                                                                                                                                                                                                                          |
| CONST_CHECKED     | CHAR(32)      |                      | <p>El byte 1 representa las restricciones de clave foránea. El byte 2 representa las restricciones de comprobación. El byte 5 representa la tabla de resumen. El byte 6 representa columnas generadas. Los demás bytes están reservados. Codifica la información de restricción en la comprobación. Valores:</p> <p>Y = Comprobado por el sistema</p> <p>U = Comprobado por el usuario</p> <p>N = No comprobado (pendiente)</p> <p>W = Estaba en un estado 'U' cuando la tabla se puso en estado pendiente de comprobación (pendiente)</p> |
| PMAP_ID           | SMALLINT      | Sí                   | Identificador del mapa de particionamiento utilizado por esta tabla. Es nulo para seudónimos y vistas.                                                                                                                                                                                                                                                                                                                                                                                                                                     |

Tabla 94. Vista de catálogo SYSCAT.TABLES (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                                                                                                                      |
|-------------------|---------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PARTITION_MODE    | CHAR(1)       |                      | <p>Modalidad utilizada para tablas de una base particionada.</p> <p>H = Información inservible en la clave de particionamiento</p> <p>R = Tabla duplicada en las particiones de base de datos</p> <p>En blanco para seudónimos, vistas y tablas en grupos de nodos de partición única sin clave de particionamiento definida. También en blanco para apodos.</p> |
| LOG_ATTRIBUTE     | CHAR(1)       |                      | <p>0 = Registro cronológico por omisión</p> <p>1 = La tabla que se ha creado inicialmente no se ha anotado cronológicamente</p>                                                                                                                                                                                                                                  |
| PCTFREE           | SMALLINT      |                      | <p>Porcentaje de cada página que se ha de reservar para inserciones posteriores. Se puede cambiar por ALTER TABLE.</p>                                                                                                                                                                                                                                           |
| APPEND_MODE       | CHAR(1)       |                      | <p>Controla cómo se insertan las filas en las páginas:</p> <p>N = Se insertan filas nuevas en espacios existentes si están disponibles</p> <p>Y = Se añaden filas nuevas al final de los datos</p> <p>Valor inicial es N.</p>                                                                                                                                    |
| REFRESH           | CHAR(1)       |                      | <p>Modalidad renovada</p> <p>D = Diferida</p> <p>I = Inmediata</p> <p>O = Una vez</p> <p>En blanco si no es una tabla de resumen</p>                                                                                                                                                                                                                             |
| REFRESH_TIME      | TIMESTAMP     | Sí                   | <p>Para REFRESH = D ó O, indicación de la hora de la sentencia REFRESH TABLE que renovó los datos por última vez. De lo contrario es nulo.</p>                                                                                                                                                                                                                   |

## SYSCAT.TABLES

Tabla 94. Vista de catálogo SYSCAT.TABLES (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                  |
|-------------------|---------------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LOCKSIZE          | CHAR(1)       |                      | Indica la granularidad de bloqueo preferente para tablas cuando son accedidas mediante sentencias DML. Sólo se aplica a tablas. Los valores posibles son:<br>R = Fila<br>T = Tabla<br>En blanco si no es aplicable<br>El valor inicial es R. |
| VOLATILE          | CHAR(1)       |                      | C = La cardinalidad de la tabla es volátil<br>En blanco si no es aplicable                                                                                                                                                                   |
| REMARKS           | VARCHAR(254)  | Sí                   | Comentario proporcionado por el usuario.                                                                                                                                                                                                     |



## SYSCAT.TABLESPACES

Contiene una fila para cada espacio de tablas.

Tabla 95. Vista de catálogo SYSCAT.TABLESPACES

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                        |
|-------------------|---------------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TBSPACE           | VARCHAR(18)   |                      | Nombre del espacio de tablas.                                                                                                                                                                      |
| DEFINER           | VARCHAR(128)  |                      | ID de autorización del encargado de definir el espacio de tablas.                                                                                                                                  |
| CREATE_TIME       | TIMESTAMP     |                      | Hora de creación del espacio de tablas.                                                                                                                                                            |
| TBSPACEID         | INTEGER       |                      | Identificador interno del espacio de tablas.                                                                                                                                                       |
| TBSPACE TYPE      | CHAR(1)       |                      | El tipo del espacio de tabla:<br>S = Espacio gestionado por el sistema<br>D = Espacio gestionado por la base de datos                                                                              |
| DATATYPE          | CHAR(1)       |                      | Tipo de datos que se puede almacenar:<br>A = Todos los tipos de datos permanentes<br>L = Sólo datos largos<br>T = Sólo tablas temporales del sistema<br>U = Sólo tablas temporales declaradas      |
| EXTENTSIZE        | INTEGER       |                      | El tamaño de la extensión, expresado en páginas de tamaño PAGESIZE. Este volumen de páginas se escribe en un contenedor individual del espacio de tablas antes de cambiar al contenedor siguiente. |
| PREFETCHSIZE      | INTEGER       |                      | Número de páginas de tamaño PAGESIZE que se deben leer cuando se efectúa una lectura anticipada.                                                                                                   |
| OVERHEAD          | DOUBLE        |                      | Actividad general del controlador y búsqueda en disco y tiempo de latencia en milisegundos.                                                                                                        |
| TRANSFERRATE      | DOUBLE        |                      | Tiempo para leer una página de tamaño PAGESIZE en el almacenamiento intermedio.                                                                                                                    |
| PAGESIZE          | INTEGER       |                      | Tamaño (en bytes) de páginas en el espacio de tablas.                                                                                                                                              |
| NGNAME            | VARCHAR(18)   |                      | Nombre del grupo de nodos para el espacio de tablas.                                                                                                                                               |
| BUFFERPOOLID      | INTEGER       |                      | ID de la agrupación de almacenamientos intermedios utilizada por este espacio de tablas (1 indica la agrupación de almacenamientos intermedios por omisión).                                       |

## SYSCAT.TABLESPACES

Tabla 95. Vista de catálogo SYSCAT.TABLESPACES (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                           |
|-------------------|---------------|----------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| DROP_RECOVERY     | CHAR(1)       |                      | N = la tabla no es recuperable después de una sentencia DROP TABLE<br>Y = la tabla es recuperable después de una sentencia DROP TABLE |
| REMARKS           | VARCHAR(254)  | Sí                   | Comentario proporcionado por el usuario.                                                                                              |

**SYSCAT.TABOPTIONS**

Cada fila contiene la opción asociada con una tabla remota.

*Tabla 96. Vista de catálogo SYSCAT.TABOPTIONS*

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                             |
|-------------------|---------------|----------------------|---------------------------------------------------------|
| TABSCHEMA         | VARCHAR(128)  |                      | Nombre calificado de tabla, vista, seudónimo o apodo.   |
| TABNAME           | VARCHAR(128)  |                      |                                                         |
| OPTION            | VARCHAR(128)  |                      | Nombre de la opción de tabla, vista, seudónimo o apodo. |
| SETTING           | VARCHAR(255)  |                      | Valor.                                                  |

## SYSCAT.TBSPACEAUTH

---

### SYSCAT.TBSPACEAUTH

Contiene una fila para cada usuario o grupo al que se ha otorgado privilegio USE para un espacio de tablas determinado de la base de datos.

Tabla 97. Vista de catálogo SYSCAT.TBSPACEAUTH

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                  |
|-------------------|---------------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GRANTOR           | CHAR(128)     |                      | ID de autorización del usuario que ha otorgado los privilegios o SYSIBM.                                                                                                                     |
| GRANTEE           | CHAR(128)     |                      | ID de autorización del usuario o grupo que tiene los privilegios.                                                                                                                            |
| GRANTEETYPE       | CHAR(1)       |                      | U = Se otorga a un usuario individual.<br>G = Se otorga a un grupo.                                                                                                                          |
| TBSPACE           | VARCHAR(18)   |                      | Nombre del espacio de tablas.                                                                                                                                                                |
| USEAUTH           | CHAR(1)       |                      | Indica si el receptor del otorgamiento tiene el privilegio USE en el espacio de tabla:<br>G = Tiene el privilegio y es otorgable.<br>N = No tiene el privilegio.<br>Y = Tiene el privilegio. |

## SYSCAT.TRIGDEP

Contiene una fila para cada dependencia que un desencadenante tiene de otro objeto.

Tabla 98. Vista de catálogo SYSCAT.TRIGDEP

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                                                                                                   |
|-------------------|---------------|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TRIGSCHEMA        | VARCHAR(128)  |                      | Nombre calificado del desencadenante.                                                                                                                                                                                                                                                                                                         |
| TRIGNAME          | VARCHAR(18)   |                      |                                                                                                                                                                                                                                                                                                                                               |
| BTYPE             | CHAR(1)       |                      | Tipo de objeto BNAME:<br>A = Seudónimo<br>F = Instancia de función<br>N = Apodo<br>O = Dependencia de privilegios en todas las subtablas o subvistas de una jerarquía de tablas o de vistas<br>R = Tipo estructurado<br>S = Tabla de resumen<br>T = Tabla<br>U = Tabla con tipo<br>V = Vista<br>W = Vista con tipo<br>X = Extensión de índice |
| BSCHEMA           | VARCHAR(128)  |                      | Nombre calificado del objeto del que depende un desencadenante.                                                                                                                                                                                                                                                                               |
| BNAME             | VARCHAR(128)  |                      |                                                                                                                                                                                                                                                                                                                                               |
| TABAUTH           | SMALLINT      | Sí                   | Si BTYPE= O, S, T, U, V o W codifica los privilegios en la tabla o vista que necesita este desencadenante; de lo contrario, es nulo.                                                                                                                                                                                                          |

## SYSCAT.TRIGGERS

---

### SYSCAT.TRIGGERS

Contiene una fila para cada desencadenante. Para jerarquías de tablas, cada desencadenante se registra sólo al nivel de la jerarquía donde se ha creado.

Tabla 99. Vista de catálogo SYSCAT.TRIGGERS

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                      |
|-------------------|---------------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TRIGSCHEMA        | VARCHAR(128)  |                      | Nombre calificado del desencadenante.                                                                                                                                                                                            |
| TRIGNAME          | VARCHAR(18)   |                      |                                                                                                                                                                                                                                  |
| DEFINER           | VARCHAR(128)  |                      | ID de autorización bajo el cual se ha definido el desencadenante.                                                                                                                                                                |
| TABSCHEMA         | VARCHAR(128)  |                      | Nombre calificado de la tabla a la que se aplica este desencadenante.                                                                                                                                                            |
| TABNAME           | VARCHAR(128)  |                      |                                                                                                                                                                                                                                  |
| TRIGTIME          | CHAR(1)       |                      | Momento en que se aplican las acciones activadas a la base de datos, en relación al suceso que ha disparado el desencadenante:<br>A = Desencadenante aplicado después del suceso<br>B = Desencadenante aplicado antes del suceso |
| TRIGEVENT         | CHAR(1)       |                      | Suceso que dispara el desencadenante.<br>I = Inserción<br>D = Supresión<br>U = Actualización                                                                                                                                     |
| GRANULARITY       | CHAR(1)       |                      | El desencadenante se ejecuta una vez por:<br>S = Sentencia<br>R = Fila                                                                                                                                                           |
| VALID             | CHAR(1)       |                      | Y = El desencadenante es válido<br>X = El desencadenante no es operativo; debe volverse a crear.                                                                                                                                 |
| CREATE_TIME       | TIMESTAMP     |                      | Hora en la que se ha definido el desencadenante. Utilizada para resolver las funciones y tipos.                                                                                                                                  |
| QUALIFIER         | VARCHAR(128)  |                      | Contiene el valor del esquema por omisión en el momento de la definición de objeto.                                                                                                                                              |
| FUNC_PATH         | VARCHAR(254)  |                      | Vía de acceso de función en el momento en que se ha definido el desencadenante. Utilizada para resolver las funciones y tipos.                                                                                                   |

Tabla 99. Vista de catálogo SYSCAT.TRIGGERS (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                               |
|-------------------|---------------|----------------------|---------------------------------------------------------------------------|
| TEXT              | CLOB(64K)     |                      | El texto completo de la sentencia CREATE TRIGGER, tal como se ha escrito. |
| REMARKS           | VARCHAR(254)  | Sí                   | Comentario suministrado por el usuario o nulo.                            |

## SYSCAT.TYEMAPPINGS

### SYSCAT.TYEMAPPINGS

Cada fila contiene una correlación definida por el usuario de un tipo de datos incorporado remoto con un tipo de datos incorporado local.

Tabla 100. Vista de catálogo SYSCAT.TYEMAPPINGS

| Nombre de columna | Tipo de datos | Posibi-<br>lidad de<br>nulos | Descripción                                                                                                                                                                              |
|-------------------|---------------|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TYPE_MAPPING      | VARCHAR(18)   |                              | Nombre de la correlación de tipos (puede ser generado por el sistema).                                                                                                                   |
| TYPESCHEMA        | VARCHAR(128)  | Sí                           | Nombre de esquema del tipo. Nulo para tipos incorporados del sistema.                                                                                                                    |
| TYPENAME          | VARCHAR(18)   |                              | Nombre del tipo local en una correlación de tipos de datos.                                                                                                                              |
| TYPEID            | SMALLINT      |                              | Identificador de tipo.                                                                                                                                                                   |
| SOURCETYPEID      | SMALLINT      |                              | Identificador de tipo de fuente.                                                                                                                                                         |
| DEFINER           | VARCHAR(128)  |                              | ID de autorización bajo el cual se ha creado esta correlación de tipos.                                                                                                                  |
| LENGTH            | INTEGER       | Sí                           | Longitud o precisión máxima del tipo de datos. Si es nulo, el sistema determina la mejor longitud/precisión.                                                                             |
| SCALE             | SMALLINT      | Sí                           | Escala para campos DECIMAL. Si es nulo, el sistema determina el mejor atributo de escala.                                                                                                |
| BIT_DATA          | CHAR(1)       | Sí                           | Y = El tipo es para datos de bits.<br>N = El tipo no es para datos de bits.<br>NULL = No se trata de un tipo de datos de caracteres o el sistema determina el atributo de datos de bits. |
| WRAPNAME          | VARCHAR(128)  | Sí                           | La correlación se aplica a este protocolo de acceso de datos.                                                                                                                            |
| SERVERNAME        | VARCHAR(128)  | Sí                           | Nombre de la fuente de datos.                                                                                                                                                            |
| SERVERTYPE        | VARCHAR (30)  | Sí                           | La correlación se aplica a este tipo de fuente de datos.                                                                                                                                 |
| SERVERVERSION     | VARCHAR(18)   | Sí                           | La correlación se aplica a esta versión de SERVERTYPE.                                                                                                                                   |
| REMOTE_TYPESCHEMA | VARCHAR(128)  | Sí                           | Nombre de esquema del tipo remoto.                                                                                                                                                       |
| REMOTE_TYPENAME   | VARCHAR(128)  |                              | Nombre del tipo de datos tal como está definido en la fuente o fuentes de datos.                                                                                                         |



Tabla 100. Vista de catálogo SYSCAT.TYPEMAPPINGS (continuación)

| Nombre de columna  | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                              |
|--------------------|---------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| REMOTE_META_TYPE   | CHAR(1)       | Sí                   | S = El tipo remoto es un tipo incorporado del sistema.<br>T = El tipo remoto es un tipo diferenciado.                                                                                    |
| REMOTE_LOWER_LEN   | INTEGER       | Sí                   | Límite inferior de la longitud/precisión del tipo decimal remoto. Para los tipos de datos de caracteres, este campo indica el número de caracteres.                                      |
| REMOTE_UPPER_LEN   | INTEGER       | Sí                   | Límite superior de la longitud/precisión del tipo decimal remoto. Para los tipos de datos de caracteres, este campo indica el número de caracteres.                                      |
| REMOTE_LOWER_SCALE | SMALLINT      | Sí                   | Límite inferior de la escala del tipo remoto.                                                                                                                                            |
| REMOTE_UPPER_SCALE | SMALLINT      | Sí                   | Límite superior de la escala del tipo remoto.                                                                                                                                            |
| REMOTE_S_OPR_P     | CHAR(2)       | Sí                   | Relación entre la escala remota y la precisión remota. Se pueden utilizar operadores de comparación básicos. Un nulo indica que no es necesaria ninguna relación específica.             |
| REMOTE_BIT_DATA    | CHAR(1)       | Sí                   | Y = El tipo es para datos de bits.<br>N = El tipo no es para datos de bits.<br>NULL = No se trata de un tipo de datos de caracteres o el sistema determina el atributo de datos de bits. |
| USER_DEFINED       | CHAR(1)       |                      | Definición suministrada por el usuario.                                                                                                                                                  |
| CREATE_TIME        | TIMESTAMP     |                      | Hora en que se ha creado la correlación.                                                                                                                                                 |
| REMARKS            | VARCHAR(254)  | Sí                   | Comentario suministrado por el usuario o nulo.                                                                                                                                           |

## SYSCAT.USEROPTIONS

---

### SYSCAT.USEROPTIONS

Cada fila contiene los valores de las opciones específicas de servidor.

*Tabla 101. Vista de catálogo SYSCAT.USEROPTIONS*

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                            |
|-------------------|---------------|----------------------|--------------------------------------------------------|
| AUTHID            | VARCHAR(128)  |                      | ID de autorización local (siempre en mayúsculas)       |
| SERVERNAME        | VARCHAR(128)  |                      | Nombre del servidor para el cual se define el usuario. |
| OPTION            | VARCHAR(128)  |                      | Nombre de las opciones de usuario.                     |
| SETTING           | VARCHAR(255)  |                      | Valor.                                                 |

## SYSCAT.VIEWDEP

Contiene una fila para cada dependencia que tiene una vista o una tabla de resumen de otro objeto. También codifica la forma en que los privilegios de esta vista dependen de los privilegios de las tablas y vistas principales.

Tabla 102. Vista de catálogo SYSCAT.VIEWDEP

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                                                                                                                                        |
|-------------------|---------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| VIEWSCHEMA        | VARCHAR(128)  |                      | Nombre de la vista o nombre de una tabla de resumen con dependencias de una tabla base.                                                                                                                                                                                                                                                                                            |
| VIEWNAME          | VARCHAR(128)  |                      |                                                                                                                                                                                                                                                                                                                                                                                    |
| DTYPE             | CHAR(1)       |                      | S = Tabla de resumen<br>V = Vista (de no tipo)<br>W = Vista con tipo                                                                                                                                                                                                                                                                                                               |
| DEFINER           | VARCHAR(128)  | Sí                   | ID de autorización del creador de la vista.                                                                                                                                                                                                                                                                                                                                        |
| BTYPE             | CHAR(1)       |                      | Tipo de objeto BNAME:<br>A = Seudónimo<br>F = Instancia de función<br>N = Apodo<br>O = Dependencia de privilegio para todas las subtablas o subvistas de una jerarquía de tablas o de vistas<br>I = Índice si se registra una dependencia de una tabla base<br>R = Tipo estructurado<br>S = Tabla de resumen<br>T = Tabla<br>U = Tabla con tipo<br>V = Vista<br>W = Vista con tipo |
| BSHEMA            | VARCHAR(128)  |                      | Nombre calificado del objeto del que depende la vista.                                                                                                                                                                                                                                                                                                                             |
| BNAME             | VARCHAR(128)  |                      |                                                                                                                                                                                                                                                                                                                                                                                    |
| TABAUTH           | SMALLINT      | Sí                   | Si BTYPE= O, S, T, U, V o W entonces codifica los privilegios en la tabla o vista subyacente en la que depende esta vista. De lo contrario es nulo.                                                                                                                                                                                                                                |

## SYSCAT.VIEWS

---

### SYSCAT.VIEWS

Contiene una o varias filas para cada vista que se crea.

Tabla 103. Vista de catálogo SYSCAT.VIEWS

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                                                                    |
|-------------------|---------------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| VIEWSCHEMA        | VARCHAR(128)  |                      | Nombre de la vista o nombre de una tabla que se utiliza para definir una tabla de resumen.                                                                                                                                                                                                                     |
| VIEWNAME          | VARCHAR(128)  |                      |                                                                                                                                                                                                                                                                                                                |
| DEFINER           | VARCHAR(128)  |                      | ID de autorización del creador de la vista.                                                                                                                                                                                                                                                                    |
| SEQNO             | SMALLINT      |                      | Siempre 1.                                                                                                                                                                                                                                                                                                     |
| VIEWCHECK         | CHAR(1)       |                      | Indica el tipo de comprobación de la vista:<br>N = Sin opción de comprobación<br>L = Opción de comprobación local<br>C = Opción de comprobación en cascada                                                                                                                                                     |
| READONLY          | CHAR(1)       |                      | Y = La vista es de sólo lectura debido a su definición.<br>N = La vista no es de sólo lectura.                                                                                                                                                                                                                 |
| VALID             | CHAR(1)       |                      | Y = La definición de una tabla de resumen o vista es válida.<br>X = La definición de una tabla de resumen o vista no es operativa; debe volverse a crear.                                                                                                                                                      |
| QUALIFIER         | VARCHAR(128)  |                      | Contiene el valor del esquema por omisión en el momento de la definición de objeto.                                                                                                                                                                                                                            |
| FUNC_PATH         | VARCHAR(254)  |                      | La vía de acceso de SQL del creador de una vista en el momento en que se ha definido la vista. Cuando se utiliza la vista en sentencias de manipulación de datos, debe utilizarse esta vía de acceso para resolver las llamadas a funciones de la vista. SYSIBM para las vistas creadas antes de la Versión 2. |
| TEXT              | CLOB(64k)     |                      | Texto de la sentencia CREATE VIEW.                                                                                                                                                                                                                                                                             |

**SYSCAT.WRAPOPTIONS**

Cada fila contiene las opciones específicas de reiniciador.

*Tabla 104. Vista de catálogo SYSCAT.WRAPOPTIONS*

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                      |
|-------------------|---------------|----------------------|----------------------------------|
| WRAPNAME          | VARCHAR(128)  |                      | Nombre de reiniciador.           |
| OPTION            | VARCHAR(128)  |                      | Nombre de opción de reiniciador. |
| SETTING           | VARCHAR(255)  |                      | Valor.                           |

## SYSCAT.WRAPPERS

---

### SYSCAT.WRAPPERS

Cada fila contiene información sobre el reiniciador registrado.

*Tabla 105. Vista de catálogo SYSCAT.WRAPPERS*

| Nombre de columna | Tipo de datos | Posibi-<br>lidad de<br>nulos | Descripción                                                                                                                 |
|-------------------|---------------|------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| WRAPNAME          | VARCHAR(128)  |                              | Nombre de reiniciador.                                                                                                      |
| WRAPTYPE          | CHAR(1)       |                              | N = No relacional<br>R = Relacional                                                                                         |
| WRAPVERSION       | INTEGER       |                              | Versión del reiniciador.                                                                                                    |
| LIBRARY           | VARCHAR(255)  |                              | Nombre del archivo que contiene el código utilizado para comunicarse con las fuentes de datos asociadas a este reiniciador. |
| REMARKS           | VARCHAR(254)  | Sí                           | Comentario suministrado por el usuario o nulo.                                                                              |

## SYSSTAT.COLDIST

Cada fila describe el valor n más frecuente o el valor cuantil n de alguna columna. Las estadísticas no se registran para columnas heredadas de tablas con tipo.

Tabla 106. Vista de catálogo SYSSTAT.COLDIST

| Nombre de columna | Tipo de datos | Posibi- lidad de nulos | Descripción                                                                                                                                                                                                                                                                                                | Actualizable |
|-------------------|---------------|------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|
| TABSCHEMA         | VARCHAR(128)  |                        | Nombre calificado de la tabla a la que se aplica esta entrada.                                                                                                                                                                                                                                             |              |
| TABNAME           | VARCHAR(128)  |                        |                                                                                                                                                                                                                                                                                                            |              |
| COLNAME           | VARCHAR(128)  |                        | Nombre de la columna a la que se aplica esta entrada.                                                                                                                                                                                                                                                      |              |
| TYPE              | CHAR(1)       |                        | Tipo de estadísticas reunidas:<br>F = Frecuencia (valor más frecuente)<br>Q = Valor cuantil                                                                                                                                                                                                                |              |
| SEQNO             | SMALLINT      |                        | Si TYPE = F, entonces N en esta columna identifica el valor N más frecuente. Si TYPE=Q, entonces N en esta columna identifica el valor N cuantil.                                                                                                                                                          |              |
| COLVALUE          | VARCHAR(254)  | Sí                     | El valor de datos, como un literal de caracteres o un valor nulo.<br><br>Esta columna se puede actualizar con una representación válida del valor adecuado para la columna con la que las estadísticas están asociadas. Si el valor de frecuencia necesario es nulo, la columna debe establecerse en NULL. | Sí           |
| VALCOUNT          | BIGINT        |                        | Si TYPE = F, entonces VALCOUNT es el número de ocurrencias de COLVALUE en la columna. Si TYPE = Q, entonces VALCOUNT es el número de filas cuyo valor es menor o igual que COLVALUE.<br><br>Esta columna sólo puede actualizarse con los valores siguientes:<br>• $\geq 0$ (cero)                          | Sí           |
| DISTCOUNT         | BIGINT        |                        | Si TYPE=q, esta columna registra el número de valores diferenciados que son menores o iguales que COLVALUE (nulo si no está disponible). el número de filas cuyo valor es menor o igual que COLVALUE.                                                                                                      | Sí           |

## SYSSTAT.COLUMNS

---

### SYSSTAT.COLUMNS

Contiene una fila para cada columna cuyas estadísticas pueden actualizarse. Las estadísticas no se registran para columnas heredadas de tablas con tipo.

Tabla 107. Vista de catálogo SYSSTAT.COLUMNS

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                                                                                                                                                                                  | Actualizable |
|-------------------|---------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|
| TABSCHEMA         | VARCHAR(128)  |                      | Nombre calificado de la tabla que contiene la columna.                                                                                                                                                                                                                                                                                                                                                                       |              |
| TABNAME           | VARCHAR(128)  |                      |                                                                                                                                                                                                                                                                                                                                                                                                                              |              |
| COLNAME           | VARCHAR(128)  |                      | Nombre de columna.                                                                                                                                                                                                                                                                                                                                                                                                           |              |
| COLCARD           | BIGINT        |                      | Número de valores diferenciados en la columna; -1 si no se reúnen estadísticas; -2 para columnas heredadas y columnas de tablas-H.<br><br>Para cualquier columna, COLCARD no puede tener un valor superior a la cardinalidad de la tabla que contiene dicha columna.<br><br>Esta columna sólo puede actualizarse con los valores siguientes: <ul style="list-style-type: none"><li>-1 ó <math>\geq 0</math> (cero)</li></ul> | Sí           |
| HIGH2KEY          | VARCHAR(33)   |                      | Segundo nivel más alto de la columna. Este campo está vacío si no se reúnen las estadísticas y para columnas heredadas y columnas de tablas-H.<br><br>Esta columna se puede actualizar con una representación válida del valor adecuado para la columna con la que las estadísticas están asociadas.<br><br>LOWKEY2 no debe ser mayor que HIGH2KEY.                                                                          | Sí           |



Tabla 107. Vista de catálogo SYSSTAT.COLUMNS (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                                                                   | Actualizable |
|-------------------|---------------|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|
| LOW2KEY           | VARCHAR(33)   | Sí                   | <p>Segundo nivel más bajo de la columna. Está vacío si no se reúnen las estadísticas y para columnas heredadas y columnas de tablas-H.</p> <p>Esta columna se puede actualizar con una representación válida del valor adecuado para la columna con la que las estadísticas están asociadas.</p>              | Sí           |
| AVGCOLLEN         | INTEGER       |                      | <p>Longitud promedio de columna. -1 si es un campo largo o LOB o bien si no se han reunido estadísticas; -2 para columnas heredadas y columnas de tablas-H.</p> <p>Esta columna sólo puede actualizarse con los valores siguientes:</p> <ul style="list-style-type: none"> <li>-1 ó &gt;= 0 (cero)</li> </ul> | Sí           |
| NUMNULLS          | BIGINT        |                      | <p>Contiene el número de nulos en una columna. -1 si no se reúnen estadísticas.</p> <p>Esta columna sólo puede actualizarse con los valores siguientes:</p> <ul style="list-style-type: none"> <li>-1 ó &gt;= 0 (cero)</li> </ul>                                                                             | Sí           |

## SYSSTAT.FUNCTIONS

---

### SYSSTAT.FUNCTIONS

Contiene una fila para cada función definida por el usuario (escalar o agregada). No incluye las funciones incorporadas. Las estadísticas no se registran para columnas heredadas de tablas con tipo.

Tabla 108. Vista de catálogo SYSSTAT.FUNCTIONS

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                             | Actualizable |
|-------------------|---------------|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|
| FUNCSHEMA         | VARCHAR(128)  |                      | Nombre de función calificado.                                                                                                                                                                                                                           |              |
| FUNCNAME          | VARCHAR(18)   |                      |                                                                                                                                                                                                                                                         |              |
| SPECIFICNAME      | VARCHAR(18)   |                      | Nombre específico de la función (instancia).                                                                                                                                                                                                            |              |
| IOS_PER_INVOC     | DOUBLE        |                      | El número estimado de E/S por invocación; -1 si no se conoce (0 valor por omisión).<br><br>Esta columna sólo puede actualizarse con los valores siguientes: <ul style="list-style-type: none"><li>• -1 ó &gt;= 0 (cero)</li></ul>                       | Sí           |
| INSTS_PER_INVOC   | DOUBLE        |                      | El número estimado de instrucciones por invocación; -1 si no se conoce (450 por omisión).<br><br>Esta columna sólo puede actualizarse con los valores siguientes: <ul style="list-style-type: none"><li>• -1 ó &gt;= 0 (cero)</li></ul>                 | Sí           |
| IOS_PER_ARGBYTE   | DOUBLE        |                      | El número estimado de E/O por byte de argumento de entrada; -1 si no se conoce (0 por omisión).<br><br>Esta columna sólo puede actualizarse con los valores siguientes: <ul style="list-style-type: none"><li>• -1 ó &gt;= 0 (cero)</li></ul>           | Sí           |
| INSTS_PER_ARGBYTE | DOUBLE        |                      | El número estimado de instrucciones por byte de argumento de entrada; -1 si no se conoce (0 por omisión).<br><br>Esta columna sólo puede actualizarse con los valores siguientes: <ul style="list-style-type: none"><li>• -1 ó &gt;= 0 (cero)</li></ul> | Sí           |

Tabla 108. Vista de catálogo SYSSTAT.FUNCTIONS (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                   | Actualizable |
|-------------------|---------------|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|
| PERCENT_ARGBYTES  | SMALLINT      |                      | El porcentaje promedio estimado de bytes de argumento de entrada que la función leerá realmente; -1 si no se conoce (100 por omisión).<br><br>Esta columna sólo puede actualizarse con los valores siguientes:<br>• -1 o entre 100 y 0 (cero) | Sí           |
| INITIAL_IOS       | DOUBLE        |                      | El número estimado de E/S realizadas la primera/última vez que se invoca la función; -1 si no se conoce (0 por omisión).<br><br>Esta columna sólo puede actualizarse con los valores siguientes:<br>• -1 ó $\geq 0$ (cero)                    | Sí           |
| INITIAL_INSTS     | DOUBLE        |                      | El número estimado de instrucciones ejecutadas la primera/última vez que se invoca la función; -1 si no se conoce (0 por omisión).<br><br>Esta columna sólo puede actualizarse con los valores siguientes:<br>• -1 ó $\geq 0$ (cero)          | Sí           |
| CARDINALITY       | BIGINT        |                      | La cardinalidad prevista de una función de tabla. -1 si no se conoce o si la función no es una función de tabla.                                                                                                                              | Sí           |
| SELECTIVITY       | DOUBLE        |                      | Utilizado para predicados definidos por el usuario. Valor por omisión = -1 si no existen predicados definidos por el usuario. Consulte la Nota 1.                                                                                             |              |

**Nota:**

1. Esta columna se establece en -1, al migrar desde la Versión 5.2 a la Versión 6.1 de DB2, en los catálogos del sistema para todas las funciones definidas por el usuario. Para un predicado definido por el usuario, la selectividad es -1 en el catálogo del sistema. En este caso, el valor de selectividad utilizado por el optimizador es 0.01.

## SYSSTAT.INDEXES

### SYSSTAT.INDEXES

Contiene una fila para cada índice que se define para una tabla.

Tabla 109. Vista de catálogo SYSSTAT.INDEXES

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                     | Actualizable |
|-------------------|---------------|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|
| INDSCHEMA         | VARCHAR(128)  |                      | Nombre calificado del índice.                                                                                                                                                   |              |
| INDNAME           | VARCHAR(18)   |                      |                                                                                                                                                                                 |              |
| TABSCHEMA         | VARCHAR(128)  |                      | Calificador del nombre de tabla.                                                                                                                                                |              |
| TABNAME           | VARCHAR(128)  |                      | Nombre de la tabla o apodo en el que se define el índice.                                                                                                                       |              |
| COLNAMES          | CLOB(1M)      |                      | Lista de nombres de columna con prefijos + o -.                                                                                                                                 |              |
| NLEAF             | INTEGER       |                      | Número de páginas; -1 si no se reúnen estadísticas.<br><br>Esta columna sólo puede actualizarse con los valores siguientes:<br>• -1 ó > 0 (cero)                                | Sí           |
| NLEVELS           | SMALLINT      |                      | Número de niveles de índices; -1 si no se reúnen estadísticas.<br><br>Esta columna sólo puede actualizarse con los valores siguientes:<br>• -1 ó > 0 (cero)                     | Sí           |
| FIRSTKEYCARD      | BIGINT        |                      | Número de valores de primera clave diferenciada; -1 si no se reúnen estadísticas.<br><br>Esta columna sólo puede actualizarse con los valores siguientes:<br>• -1 ó >= 0 (cero) | Sí           |

Tabla 109. Vista de catálogo SYSSTAT.INDEXES (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                            | Actualizable |
|-------------------|---------------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|
| FIRST2KEYCARD     | BIGINT        |                      | Número de claves diferenciadas que utilizan las dos primeras columnas del índice (-1 si no hay estadísticas o no son aplicables)<br><br>Esta columna sólo puede actualizarse con los valores siguientes:<br>• -1 ó $\geq 0$ (cero)     | Sí           |
| FIRST3KEYCARD     | BIGINT        |                      | Número de claves diferenciadas que utilizan las tres primeras columnas del índice (-1 si no hay estadísticas o si no son aplicables)<br><br>Esta columna sólo puede actualizarse con los valores siguientes:<br>• -1 ó $\geq 0$ (cero) | Sí           |
| FIRST4KEYCARD     | BIGINT        |                      | Número de claves diferenciadas que utilizan las cuatro primeras columnas del índice (-1 si no hay estadísticas o no son aplicables)<br><br>Esta columna sólo puede actualizarse con los valores siguientes:<br>• -1 ó $\geq 0$ (cero)  | Sí           |
| FULLKEYCARD       | BIGINT        |                      | Número de valores de clave completa diferenciada; -1 si no se reúnen estadísticas.<br><br>Esta columna sólo puede actualizarse con los valores siguientes:<br>• -1 ó $\geq 0$ (cero)                                                   | Sí           |

## SYSSTAT.INDEXES

Tabla 109. Vista de catálogo SYSSTAT.INDEXES (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                                                                                                          | Actualizable |
|-------------------|---------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|
| CLUSTERRATIO      | SMALLINT      |                      | <p>La utiliza el optimizador. Indica el grado de agrupación de los datos que contienen el índice; -1 si no se reúnen estadísticas o si se han reunido estadísticas de índice detalladas.</p> <p>Esta columna sólo puede actualizarse con los valores siguientes:</p> <ul style="list-style-type: none"> <li>-1 ó entre 0 y 100</li> </ul>            | Sí           |
| CLUSTERFACTOR     | DOUBLE        |                      | <p>La utiliza el optimizador. Es una mejor medición del grado de agrupación ó -1 si no se han reunido estadísticas de índice detalladas.</p> <p>Esta columna sólo puede actualizarse con los valores siguientes:</p> <ul style="list-style-type: none"> <li>-1 ó entre 0 y 1</li> </ul>                                                              | Sí           |
| SEQUENTIAL_PAGES  | INTEGER       |                      | <p>Número de páginas ubicadas en disco por orden de clave de índice con pocos o ningún vacío entre ellas. (-1 si no hay estadísticas disponibles.)</p> <p>Esta columna sólo puede actualizarse con los valores siguientes:</p> <ul style="list-style-type: none"> <li>-1 ó &gt;= 0 (cero)</li> </ul>                                                 | Sí           |
| DENSITY           | INTEGER       |                      | <p>Proporción de SEQUENTIAL_PAGES para numerar las páginas del rango de páginas ocupadas por el índice, expresada como un porcentaje (entero entre 0 y 100, -1 si no hay estadísticas disponibles.)</p> <p>Esta columna sólo puede actualizarse con los valores siguientes:</p> <ul style="list-style-type: none"> <li>-1 ó entre 0 y 100</li> </ul> | Sí           |

Tabla 109. Vista de catálogo SYSSTAT.INDEXES (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Actualizable |
|-------------------|---------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|
| PAGE_FETCH_PAIRS  | VARCHAR(254)  |                      | <p>Una lista de pares de enteros, representada en la forma de caracteres. Cada par representa el número de páginas de un almacenamiento intermedio hipotético, el número de lecturas de páginas necesario para explorar el índice utilizando dicho almacenamiento hipotético. (Serie de longitud cero si no hay datos disponibles.)</p> <p>Esta columna puede actualizarse con los siguientes valores de entrada:</p> <ul style="list-style-type: none"> <li>• El delimitador de pares y los caracteres separadores de pares son sólo caracteres no numéricos aceptados.</li> <li>• Los blancos sólo son caracteres reconocidos como delimitador de pares y separador de pares.</li> <li>• Cada entrada numérica debe tener una entrada numérica asociada que le acompañe, separadas las dos por el carácter separador de pares.</li> <li>• Cada par debe estar separado de los otros pares por el carácter delimitador de pares.</li> <li>• Cada entrada de número esperada debe estar entre 0 y 9 (sólo valores positivos).</li> </ul> | Sí           |

## SYSSTAT.TABLES

---

### SYSSTAT.TABLES

Contiene una fila para cada tabla *base*. Por lo tanto, las vistas o seudónimos no se incluyen. Para las tablas con tipo, sólo se incluye la tabla raíz de una jerarquía de tablas en esta vista. Las estadísticas no se registran para columnas heredadas de tablas con tipo. El valor CARD se aplica a la tabla raíz solamente mientras las otras estadísticas se apliquen a toda la jerarquía de tablas.

Tabla 110. Vista de catálogo SYSSTAT.TABLES

| Nombre de columna | Tipo de datos | Posibi-<br>lidad<br>de nulos | Descripción                                                                                                                                                                                                                                                                                                                                                                           | Actualizable |
|-------------------|---------------|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|
| TABSHEMA          | VARCHAR(128)  |                              | Nombre calificado de la tabla.                                                                                                                                                                                                                                                                                                                                                        |              |
| TABNAME           | VARCHAR(128)  |                              |                                                                                                                                                                                                                                                                                                                                                                                       |              |
| CARD              | BIGINT        |                              | Número total de filas en la tabla; -1 si no se reúnen estadísticas.<br><br>Una actualización a CARD para una tabla no debe intentar asignarle un valor menor que el valor COLCARD de cualquiera de las columnas de dicha tabla. Esta columna sólo puede actualizarse con los valores siguientes: <sup>116</sup> <ul style="list-style-type: none"><li>• -1 ó &gt;= 0 (cero)</li></ul> | Sí           |
| NPAGES            | INTEGER       |                              | El número total de páginas en las que las filas de la tabla existen; -1 si no se reúnen las estadísticas; -2 para subtablas y tablas-H.<br><br>Esta columna sólo puede actualizarse con los valores siguientes: <sup>116</sup> <ul style="list-style-type: none"><li>• -1 ó &gt;= 0 (cero)</li></ul>                                                                                  | Sí           |
| FPAGES            | INTEGER       |                              | El número total de páginas en el archivo; -1 si no se reúnen las estadísticas ; -2 para subtablas y tablas-H.<br><br>Esta columna sólo puede actualizarse con los valores siguientes: <sup>116</sup> <ul style="list-style-type: none"><li>• -1 ó &gt;= 0 (cero)</li></ul>                                                                                                            | Sí           |

---

116. No puede cambiarse un valor de -2 y no puede establecerse directamente un valor de columna a -2.



Tabla 110. Vista de catálogo SYSSTAT.TABLES (continuación)

| Nombre de columna | Tipo de datos | Posibi-<br>lidad<br>de nulos | Descripción                                                                                                                                                                                                                                                                                                    | Actualizable |
|-------------------|---------------|------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|
| OVERFLOW          | INTEGER       |                              | <p>El número total de registros de desbordamiento en la tabla; -1 si no se reúnen las estadísticas; -2 para subtablas y tablas-H.</p> <p>Esta columna sólo puede actualizarse con los valores siguientes: <sup>116</sup></p> <ul style="list-style-type: none"> <li>-1 ó <math>\geq 0</math> (cero)</li> </ul> | Sí           |

## SYSSTAT.TABLES

---

## Apéndice E. Vistas de catálogo para utilizar con tipos estructurados

Cuando se utilizan índices ampliados, existen vistas adicionales que proporcionan información útil que sirve de complemento a la vista de catálogo SYSCAT. Estas vistas no se crean automáticamente. Las vistas se crean en el esquema OBJCAT y para todas ellas se otorga por omisión el privilegio SELECT a PUBLIC.

**AVISO:** Este conjunto de vistas sólo es para uso temporal hasta que aparezca la próxima versión que dé soporte a la migración de catálogos. Las aplicaciones no deben suponer que estas vistas existen en todas las bases de datos y deben considerar que estas vistas de catálogo pueden dejar de existir en versiones futuras. La información proporcionada por estas vistas estará disponible a través de las vistas SYSCAT en una versión futura.

Las vistas pueden crearse siguiendo estos pasos:

- Mediante el Procesador de línea de mandatos, conéctese a la base de datos con un ID de autorización que tenga autorización SYSADM o DBADM.
- Asegúrese de que está en el directorio inicial de la instancia de DB2.
- Si utiliza un sistema basado en UNIX, emita este mandato:  

```
db2 -tvf sqllib/bin/objcat.db2
```
- Si utiliza un sistema OS/2 o basado en Windows emita este mandato:  

```
db2 -tvf sqllib\bin\objcat.db2
```

Las vistas creadas por **objcat.db2** se pueden eliminar siguiendo estos pasos:

- Mediante el Procesador de línea de mandatos, conéctese a la base de datos con un ID de autorización que tenga autorización SYSADM o DBADM.
- Asegúrese de que está en el directorio inicial de la instancia de DB2.
- Si utiliza un sistema basado en UNIX, emita este mandato:  

```
db2 -tvf sqllib/bin/objcatdp.db2
```
- Si utiliza un sistema OS/2 o basado en Windows emita este mandato:  

```
db2 -tvf sqllib\bin\objcatdp.db2
```

**Nota:** Si la base de datos ya incluye un esquema llamado OBJCAT, puede ser necesario que haga su propia copia del archivo **objcat.db2** y cambie los nombres de esquema en la segunda y tercera sentencia CREATE SCHEMA a un nombre apropiado.

Las sentencias del archivo OBJCAT.DB2 crearán vista de catálogo OBJCAT adicionales.

Este apéndice contiene una descripción de las vistas de catálogo OBJCAT. Para conocer las vistas SYSCAT asociadas, vea el “Apéndice D. Vistas de catálogo” en la página 1281.

Las vistas de catálogo se actualizan durante la actividad normal, en respuesta a sentencias de definición de datos SQL, rutinas de entorno y determinados programas de utilidad. Los datos de las vistas de catálogo son accesibles mediante las funciones normales de consulta del SQL. Las columnas tienen nombres uniformes basados en el tipo de objetos que describen:

| <b>Objeto descrito</b>                           | <b>Nombres de columna</b>    |
|--------------------------------------------------|------------------------------|
| <b>Tabla</b>                                     | TABSCHEMA, TABNAME           |
| <b>Índice</b>                                    | INDSCHEMA, INDNAME           |
| <b>Vista</b>                                     | VIEWSCHEMA, VIEWNAME         |
| <b>Restricción</b>                               | CONSTSCHEMA, CONSTNAME       |
| <b>Desencadenante</b>                            | TRIGSCHEMA, TRIGNAME         |
| <b>Paquete</b>                                   | PKGSCHEMA, PKGNAME           |
| <b>Tipo</b>                                      | TYPESCHEMA, TYPENAME, TYPEID |
| <b>Función</b>                                   | FUNCSCHEMA, FUNCNAME, FUNCID |
| <b>Columna</b>                                   | COLNAME                      |
| <b>Atributo</b>                                  | ATTR_NAME                    |
| <b>Esquema</b>                                   | SCHEMANAME                   |
| <b>Espacio de tablas</b>                         | TBSPACE                      |
| <b>Grupo de nodos</b>                            | NGNAME                       |
| <b>Agrupación de almacenamientos intermedios</b> | BPNAME                       |
| <b>Supervisor de sucesos</b>                     | EVMONNAME                    |
| <b>Indicación de la hora de creación</b>         | CREATE_TIME                  |

---

## 'Guía' de las vistas de catálogo

| Descripción                         | Vista de catálogo            | Página |
|-------------------------------------|------------------------------|--------|
| índices                             | OBJCAT.INDEXES               | 1400   |
| reglas de explotación de índice     | OBJCAT.INDEXEXPLOITRULES     | 1404   |
| dependencias de extensión de índice | OBJCAT.INDEXEXTENSIONDEP     | 1405   |
| métodos de extensión de índice      | OBJCAT.INDEXEXTENSIONMETHODS | 1406   |
| parámetros de extensión de índice   | OBJCAT.INDEXEXTENSIONPARMS   | 1407   |
| extensiones de índice               | OBJCAT.INDEXEXTENSIONS       | 1408   |
| especificaciones de predicado       | OBJCAT.PREDICATESPECS        | 1409   |
| transformaciones                    | OBJCAT.TRANSFORMS            | 1410   |

## OBJCAT.INDEXES

---

### OBJCAT.INDEXES

Tabla 111. Vista de catálogo OBJCAT.INDEXES

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                                                        |
|-------------------|---------------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| INDSCHEMA         | VARCHAR(128)  |                      | Nombre del índice.                                                                                                                                                                                                                                                                                 |
| INDNAME           | VARCHAR(18)   |                      |                                                                                                                                                                                                                                                                                                    |
| DEFINER           | VARCHAR(128)  |                      | Usuario que ha creado el índice.                                                                                                                                                                                                                                                                   |
| TABSCHEMA         | VARCHAR(128)  |                      | Nombre calificado de la tabla o apodo en el que se define el índice.                                                                                                                                                                                                                               |
| TABNAME           | VARCHAR(128)  |                      |                                                                                                                                                                                                                                                                                                    |
| COLNAMES          | VARCHAR(640)  |                      | Lista de nombres de columnas, cada uno precedido por un signo + o – para indicar orden ascendente o descendente respectivamente. Aviso: Esta columna se eliminará en el futuro. Utilice “SYSCAT.INDEXCOLUSE” en la página 1326 para obtener esta información.                                      |
| UNIQUERULE        | CHAR(1)       |                      | Regla de unicidad:<br>D = Los duplicados están permitidos<br>P = Índice primario<br>U = Sólo se permiten entradas exclusivas                                                                                                                                                                       |
| MADE_UNIQUE       | CHAR(1)       |                      | Y = El índice era inicialmente un índice de no unicidad, pero se ha convertido a un índice de unicidad para dar soporte a una restricción de unicidad o de clave primaria. Si se elimina la restricción, el índice volverá a ser de no unicidad.<br>N = El índice permanece tal como se ha creado. |
| COLCOUNT          | SMALLINT      |                      | Número de columnas de la clave más el número de columnas INCLUDE, si hay alguna.                                                                                                                                                                                                                   |
| UNIQUE_COLCOUNT   | SMALLINT      |                      | Número de columnas necesarias para una clave de unicidad. Es siempre menor o igual que COLCOUNT. Es menor que COLCOUNT sólo si hay columnas INCLUDE. -1 si el índice no tiene clave de unicidad (permite los duplicados).                                                                          |
| INDEXTYPE         | CHAR(4)       |                      | Tipo de índice.<br>CLUS = Agrupación<br>REG = Normal                                                                                                                                                                                                                                               |

Tabla 111. Vista de catálogo OBJCAT.INDEXES (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                       |
|-------------------|---------------|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ENTRYTYPE         | CHAR(1)       |                      | H = Un índice en una tabla de jerarquía (tabla-H)<br>L = Índice lógico en una tabla con tipo en blanco si es un índice en una tabla de no tipo                                                    |
| PCTFREE           | SMALLINT      |                      | Porcentaje de cada página de hoja de índice que se va a reservar durante la creación inicial del índice. Este espacio está disponible para inserciones futuras después de la creación del índice. |
| IID               | SMALLINT      |                      | ID interno de índice                                                                                                                                                                              |
| NLEAF             | INTEGER       |                      | Número de páginas de índice; -1 si no se recogen datos estadísticos.                                                                                                                              |
| NLEVELS           | SMALLINT      |                      | Número de niveles de índice; -1 si no se recogen datos estadísticos.                                                                                                                              |
| FIRSTKEYCARD      | BIGINT        |                      | Número de valores de primera clave diferenciada (-1 si no se recogen datos estadísticos).                                                                                                         |
| FIRST2KEYCARD     | BIGINT        |                      | Número de claves diferenciadas que utilizan las dos primeras columnas del índice (-1 si no se recogen datos estadísticos o si no es aplicable).                                                   |
| FIRST3KEYCARD     | BIGINT        |                      | Número de claves diferenciadas que utilizan las tres primeras columnas del índice (-1 si no se recogen datos estadísticos o si no es aplicable).                                                  |
| FIRST4KEYCARD     | BIGINT        |                      | Número de claves diferenciadas que utilizan las cuatro primeras columnas del índice (-1 si no se recogen datos estadísticos o si no es aplicable).                                                |
| FULLKEYCARD       | BIGINT        |                      | Número de valores de clave diferenciada completa; -1 si no se recogen datos estadísticos.                                                                                                         |
| CLUSTERRATIO      | SMALLINT      |                      | Grado de agrupación de los datos con el índice; -1 si no se recogen datos estadísticos o si se recogen estadísticas de índice detalladas (en cuyo caso, se utilizará CLUSTERFACTOR en su lugar).  |

## OBJCAT.INDEXES

Tabla 111. Vista de catálogo OBJCAT.INDEXES (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                                                                                                                                                                 |
|-------------------|---------------|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CLUSTERFACTOR     | DOUBLE        |                      | Medición más refinada del grado de agrupación o -1 si no se han recogido estadísticas de índice detalladas o si el índice está definido en un apodo.                                                                                                                                                                                                                                                        |
| SEQUENTIAL_PAGES  | INTEGER       |                      | Número de páginas ubicadas en disco por orden de clave de índice, con pocos o ningún hueco entre ellas (-1 si no hay datos estadísticos).                                                                                                                                                                                                                                                                   |
| DENSITY           | INTEGER       |                      | Relación entre SEQUENTIAL_PAGES y el número de páginas del rango de páginas ocupadas por el índice, expresada como porcentaje (entero entre 0 y 100, -1 si no hay datos estadísticos disponibles).                                                                                                                                                                                                          |
| USER_DEFINED      | SMALLINT      |                      | 1 si un usuario ha definido este índice y no se ha eliminado; de lo contrario, 0.                                                                                                                                                                                                                                                                                                                           |
| SYSTEM_REQUIRED   | SMALLINT      |                      | 1 si este índice es necesario para la restricción de clave primaria o de clave de unicidad O BIEN si es el índice para la columna de identificador de objeto (OID) de una tabla con tipo.<br><br>2 si este índice es necesario para la restricción de clave primaria o de clave de unicidad Y es el índice en la columna de identificador de objeto (OID) de una tabla con tipo.<br><br>De lo contrario, 0. |
| CREATE_TIME       | TIMESTAMP     |                      | Hora en que se ha creado el índice.                                                                                                                                                                                                                                                                                                                                                                         |
| STATS_TIME        | TIMESTAMP     | Sí                   | Última vez que se ha realizado un cambio en las estadísticas registradas para este índice. Nulo si no hay estadísticas disponibles.                                                                                                                                                                                                                                                                         |
| PAGE_FETCH_PAIRS  | VARCHAR(254)  |                      | Una lista de pares de enteros, representada en la forma de caracteres. Cada par representa el número de páginas de un almacenamiento intermedio hipotético y el número de lecturas de páginas necesario para explorar la tabla con este índice utilizando dicho almacenamiento intermedio hipotético. (Serie de longitud cero si no hay datos disponibles).                                                 |



Tabla 111. Vista de catálogo OBJCAT.INDEXES (continuación)

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                          |
|-------------------|---------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| MINPCTUSED        | SMALLINT      |                      | Si no es cero, se habilita la reorganización del índice en línea y el valor es el umbral del espacio mínimo utilizado antes de fusionar las páginas. |
| REVERSE_SCANS     | CHAR(1)       |                      | Y = El índice soporta exploraciones invertidas<br>N = El índice no soporta exploraciones invertidas                                                  |
| INTERNAL_FORMAT   | SMALLINT      |                      | Codifica la representación interna del índice.                                                                                                       |
| IESHEMA           | VARCHAR(128)  | Sí                   | Nombre calificado de la extensión de índice.                                                                                                         |
| IENAME            | VARCHAR(18)   | Sí                   | Nulo para índices ordinarios.                                                                                                                        |
| IEARGUMENTS       | CLOB(32K)     | Sí                   | Información externa del parámetro especificado cuando se crea el índice. Nulo para índices ordinarios.                                               |
| REMARKS           | VARCHAR(254)  | Sí                   | Comentario suministrado por el usuario o nulo.                                                                                                       |

## OBJCAT.INDEXEXPLOITRULES

---

### OBJCAT.INDEXEXPLOITRULES

Cada fila representa una explotación de índice.

Tabla 112. Vista de catálogo OBJCAT.INDEXEXPLOITRULES

| Nombre de columna | Tipo de datos | Posibi-<br>lidad de<br>nulos | Descripción                                                                      |
|-------------------|---------------|------------------------------|----------------------------------------------------------------------------------|
| FUNCID            | INTEGER       |                              | ID de función.                                                                   |
| SPECID            | SMALLINT      |                              | Número de la especificación de predicado dentro de la sentencia CREATE FUNCTION. |
| IESHEMA           | VARCHAR(128)  |                              | Nombre calificado de la extensión de índice.                                     |
| IENAME            | VARCHAR(18)   |                              |                                                                                  |
| RULEID            | SMALLINT      |                              | ID exclusivo de la regla de explotación.                                         |
| SEARCHMETHODID    | SMALLINT      |                              | ID del método de búsqueda en la extensión de índice específica.                  |
| SEARCHKEY         | VARCHAR(320)  |                              | Clave utilizada para explotar el índice.                                         |
| SEARCHARGUMENT    | VARCHAR(1800) |                              | Argumentos de búsqueda utilizados en la explotación de índice.                   |

**OBJCAT.INDEXEXTENSIONDEP**

Contiene una fila para cada dependencia que las extensiones de índice tienen sobre diversos objetos de base de datos.

Tabla 113. Vista de catálogo OBJCAT.INDEXEXTENSIONDEP

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                                                                                                                                                                                                                                                     |
|-------------------|---------------|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IESHEMA           | VARCHAR(128)  |                      | Nombre calificado de la extensión de índice que tiene dependencias sobre otro objeto.                                                                                                                                                                                                                                                                                                           |
| IENAME            | VARCHAR(18)   |                      |                                                                                                                                                                                                                                                                                                                                                                                                 |
| BTYPE             | CHAR(1)       |                      | Tipo de objeto del que depende la extensión de índice:<br>A = Seudónimo<br>F = Instancia de función o instancia de método<br>J = Definición de servidor<br>O = Dependencia "externa" sobre privilegio SELECT jerárquico<br>R = Tipo estructurado<br>S = Tabla de resumen<br>T = Tabla (sin tipo)<br>U = Tabla con tipo<br>V = Vista (sin tipo)<br>W = Vista con tipo<br>X = Extensión de índice |
| BSCHEMA           | VARCHAR(128)  |                      | Nombre calificado del objeto del que depende la extensión de índice (si BTYPE='F', este campo es el nombre específico de una función).                                                                                                                                                                                                                                                          |
| BNAME             | VARCHAR(128)  |                      |                                                                                                                                                                                                                                                                                                                                                                                                 |
| TABAUTH           | SMALLINT      | Sí                   | Si BTYPE='O', 'T', 'U', 'V' o 'W', codifica los privilegios que un desencadenante dependiente necesita para la tabla (o vista); en otro caso su valor es nulo.                                                                                                                                                                                                                                  |

## OBJCAT.INDEXEXTENSIONMETHODS

---

### OBJCAT.INDEXEXTENSIONMETHODS

Cada fila representa un método de búsqueda. Una extensión de índice puede contener varios métodos de búsqueda.

Tabla 114. Vista de catálogo OBJCAT.INDEXEXTENSIONMETHODS

| Nombre de columna  | Tipo de datos | Posibi-<br>lidad de<br>nulos | Descripción                                  |
|--------------------|---------------|------------------------------|----------------------------------------------|
| METHODNAME         | VARCHAR(18)   |                              | Nombre del método de búsqueda.               |
| METHODID           | SMALLINT      |                              | Número del método en la extensión de índice. |
| IESHEMA            | VARCHAR(128)  |                              | Nombre calificado de la extensión de índice. |
| IENAME             | VARCHAR(18)   |                              |                                              |
| RANGEFUNCSHEMA     | VARCHAR(128)  |                              | Nombre calificado de la función de rangos.   |
| RANGEFUNCNAME      | VARCHAR(18)   |                              |                                              |
| RANGESPECIFICNAME  | VARCHAR(18)   |                              | Nombre específico de la función de rangos.   |
| FILTERFUNCSHEMA    | VARCHAR(128)  |                              | Nombre calificado de la función de filtro.   |
| FILTERFUNCNAME     | VARCHAR(18)   |                              |                                              |
| FILTERSPECIFICNAME | VARCHAR(18)   |                              | Nombre específico de la función de filtro.   |
| REMARKS            | VARCHAR(254)  | Sí                           | Proporcionado por el usuario o nulo.         |

**OBJCAT.INDEXEXTENSIONPARMS**

Cada fila representa un parámetro de instancia de la extensión de índice o una definición de la clave fuente.

Tabla 115. Vista de catálogo OBJCAT.INDEXEXTENSIONPARMS

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                   |
|-------------------|---------------|----------------------|---------------------------------------------------------------------------------------------------------------|
| IESHEMA           | VARCHAR(128)  |                      | Nombre calificado de la extensión de índice.                                                                  |
| IENAME            | VARCHAR(18)   |                      |                                                                                                               |
| ORDINAL           | SMALLINT      |                      | Número de orden del parámetro o clave fuente.                                                                 |
| PARMNAME          | VARCHAR(18)   |                      | Nombre del parámetro o clave fuente.                                                                          |
| TYPESHEMA         | VARCHAR(128)  |                      | Nombre calificado del parámetro de instancia o del tipo de datos de la clave fuente.                          |
| TYPENAME          | VARCHAR(18)   |                      |                                                                                                               |
| LENGTH            | INTEGER       |                      | Longitud del parámetro de instancia o del tipo de datos de la clave fuente.                                   |
| SCALE             | SMALLINT      |                      | Escala del parámetro de instancia o del tipo de datos de la clave fuente. Igual a 0 si no es aplicable.       |
| PARMTYPE          | CHAR(1)       |                      | Tipo representado por la fila:<br>P = parámetro de extensión de índice<br>K = columna de clave                |
| CODEPAGE          | SMALLINT      |                      | Página de códigos del parámetro de extensión de índice. Es igual a 0 si no es un tipo de serie de caracteres. |

## OBJCAT.INDEXEXTENSIONS

---

### OBJCAT.INDEXEXTENSIONS

Contiene una fila para cada extensión de índice.

Tabla 116. Vista de catálogo OBJCAT.INDEXEXTENSIONS

| Nombre de columna  | Tipo de datos | Posibi-<br>lidad de<br>nulos | Descripción                                                           |
|--------------------|---------------|------------------------------|-----------------------------------------------------------------------|
| IESHEMA            | VARCHAR(128)  |                              | Nombre calificado de la extensión de índice.                          |
| IENAME             | VARCHAR(18)   |                              |                                                                       |
| DEFINER            | VARCHAR(128)  |                              | ID de autorización con el cual se ha definido la extensión de índice. |
| CREATE_TIME        | TIMESTAMP     |                              | Hora en la que se definió la extensión de índice.                     |
| KEYGENFUNCSHEMA    | VARCHAR(128)  |                              | Nombre calificado de la función de generación de claves.              |
| KEYGENFUNCNAME     | VARCHAR(18)   |                              |                                                                       |
| KEYGENSPECIFICNAME | VARCHAR(18)   |                              | Nombre específico de la función de generación de claves.              |
| TEXT               | CLOB(64K)     |                              | El texto completo de la sentencia CREATE INDEX EXTENSION.             |
| REMARKS            | VARCHAR(254)  |                              | Comentario suministrado por el usuario o nulo.                        |

**OBJCAT.PREDICATESPECS**

Tabla 117. Vista de catálogo OBJCAT.PREDICATESPECS

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                   |
|-------------------|---------------|----------------------|-----------------------------------------------------------------------------------------------|
| FUNCSHEMA         | VARCHAR(128)  |                      | Nombre calificado de la función.                                                              |
| FUNCNAME          | VARCHAR(18)   |                      |                                                                                               |
| SPECIFICNAME      | VARCHAR(18)   |                      | El nombre de la instancia de función.                                                         |
| FUNCID            | INTEGER       |                      | ID de función.                                                                                |
| SPECID            | SMALLINT      |                      | ID de la especificación de predicado.                                                         |
| CONTEXTOP         | CHAR(8)       |                      | El operador de comparación es uno de los operadores relacionales incorporados (=,<,>=, etc.). |
| CONTEXTEXP        | CLOB(32K)     |                      | Constante o una expresión SQL.                                                                |
| FILTERTEXT        | CLOB(32K)     | Sí                   | Texto de la expresión de filtro de datos.                                                     |

## OBJCAT.TRANSFORMS

### OBJCAT.TRANSFORMS

Contiene una fila para cada tipo de función de transformación dentro de un tipo definido por el usuario contenido en un grupo de transformación especificado.

Tabla 118. Vista de catálogo OBJCAT.TRANSFORMS

| Nombre de columna | Tipo de datos | Posibilidad de nulos | Descripción                                                                                                                                                          |
|-------------------|---------------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TYPEID            | SMALLINT      |                      | ID interno de tipo, tal como está definido en SYSCAT.DATATYPES                                                                                                       |
| TYPESHEMA         | VARCHAR(128)  |                      | Nombre calificado del tipo estructurado proporcionado, definido por el usuario.                                                                                      |
| TYPENAME          | VARCHAR(18)   |                      |                                                                                                                                                                      |
| GROUPNAME         | VARCHAR(18)   |                      | Nombre del grupo de transformación.                                                                                                                                  |
| FUNCID            | INTEGER       | Sí                   | ID interno de la función de transformación asociada, tal como está definido en SYSCAT.FUNCTIONS. Nulo sólo para funciones internas del sistema.                      |
| FUNCHEMA          | VARCHAR(128)  |                      | Nombre calificado de las funciones de transformación asociadas.                                                                                                      |
| FUNCNAME          | VARCHAR(18)   |                      |                                                                                                                                                                      |
| SPECIFICNAME      | VARCHAR(18)   |                      | Nombre específico de la función (instancia).                                                                                                                         |
| TRANSFORMTYPE     | VARCHAR(8)    |                      | 'FROM SQL' = La función de transformación transforma un tipo estructurado desde SQL<br>'TO SQL' = La función de transformación transforma un tipo estructurado a SQL |
| FORMAT            | CHAR(1)       |                      | 'U' = Definido por el usuario                                                                                                                                        |
| MAXLENGTH         | INTEGER       | Sí                   | Longitud máxima (en bytes) de la salida de la transformación FROM SQL. Nulo para las transformaciones TO SQL.                                                        |
| ORIGIN            | CHAR(1)       |                      | 'I' = Heredado en la jerarquía de tipos.<br>'U' = Definido por el usuario.                                                                                           |
| REMARKS           | VARCHAR(254)  | Sí                   | Comentario proporcionado por el usuario o nulo.                                                                                                                      |



---

## Apéndice F. Sistemas federados

Este apéndice documenta:

- Los tipos de servidor que pueden definirse en las sentencias de SQL para establecer y utilizar sistemas federados DB2.
- Las opciones que pueden definirse en las sentencias de SQL para establecer y utilizar sistemas federados DB2.
- Las correlaciones por omisión entre los tipos de datos soportados por el servidor federado y los tipos de datos soportados por las fuentes de datos.
- Factores a tener en cuenta y restricciones a observar al utilizar el paso a través.

---

### Tipos de servidor

Los tipos de servidor indican la clase de fuente de datos a la que el servidor representará. Los tipos de servidor varían según el proveedor, la finalidad y la plataforma. Los valores soportados varían según el reiniciador utilizado.

- **Reiniciador DRDA**

Familia DB2

*Tabla 119. IBM DB2 Universal Database*

| Tipo de servidor | Fuente de datos                     |
|------------------|-------------------------------------|
| DB2/UDB          | IBM DB2 Universal Database          |
| DataJoiner       | IBM DB2 DataJoiner V2.1 y V2.1.1    |
| DB2/6000         | IBM DB2 para AIX                    |
| DB2/HPUX         | IBM DB2 para HP-UX V1.2             |
| DB2/NT           | IBM DB2 para Windows NT             |
| DB2/EEE          | IBM DB2 Enterprise-Extended Edition |
| DB2/SUN          | IBM DB2 para Solaris V1 y V1.2      |
| DB2/2            | IBM DB2 para OS/2                   |
| DB2/LINUX        | IBM DB2 para Linux                  |
| DB2/PTX          | IBM DB2 para NUMA-Q                 |
| DB2/SCO          | IBM DB2 para SCO Unixware           |

*Tabla 120. IBM DB2 Universal Database para AS/400*

| Tipo de servidor | Fuente de datos     |
|------------------|---------------------|
| DB2/400          | IBM DB2 para AS/400 |

Tabla 121. IBM DB2 Universal Database para OS/390

| Tipo de servidor | Fuente de datos     |
|------------------|---------------------|
| DB2/390          | IBM DB2 para OS/390 |
| DB2/MVS          | IBM DB2 para MVS    |

Tabla 122. IBM DB2 Server para VM y VSE

| Tipo de servidor | Fuente de datos  |
|------------------|------------------|
| DB2/VM           | IBM DB2 para VM  |
| DB2/VSE          | IBM DB2 para VSE |
| SQL/DS           | IBM SQL/DS       |

- **Reiniciador SQLNET**

Fuentes de datos Oracle soportadas por software de cliente Oracle SQL\*Net V1 o V2.

| Tipo de servidor | Fuente de datos            |
|------------------|----------------------------|
| ORACLE           | Oracle V7.0.13 o posterior |

- **Reiniciador NET8**

Fuentes de datos Oracle soportadas por software de cliente Oracle Net8.

| Tipo de servidor | Fuente de datos            |
|------------------|----------------------------|
| ORACLE           | Oracle V7.0.13 o posterior |

- **Reiniciador OLE DB**

Proveedores OLE DB compatibles con Microsoft OLE DB 2.0 o posterior.

| Tipo de servidor | Fuente de datos            |
|------------------|----------------------------|
| -                | Cualquier proveedor OLE DB |

- **Otros reiniciadores**

Consulte la documentación incluida con el reiniciador.

## Opciones SQL para sistemas federados

Esta sección describe:

- Las opciones de columna que se pueden especificar en la sentencia ALTER NICKNAME.
- Las opciones de correlación de funciones que se pueden especificar en la sentencia CREATE FUNCTION MAPPING.

- Las opciones del servidor que se pueden especificar en las sentencias CREATE SERVER, ALTER SERVER y SET SERVER OPTION.
- Las opciones de usuario que se pueden especificar en las sentencias CREATE USER MAPPING y ALTER USER MAPPING.

## Opciones de columna

La finalidad principal de las opciones de columna es proporcionar información acerca de las columnas de apodo al compilador de SQL. Cuando se establecen las opciones de columna para una o varias columnas en 'Y' se permite al compilador tomar en consideración las posibilidades de bajada adicionales para predicados que realizan operaciones de evaluación. Consulte el manual *Administration Guide: Performance* para obtener más información sobre el proceso de bajada.

Tabla 123. Opciones de columna y sus valores

| Opción                                                                                                                                                                                                                                                                                                                                                                                                                                          | Valores válidos | Valor por omisión                                                                                                                                                                    |     |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| numeric_string                                                                                                                                                                                                                                                                                                                                                                                                                                  | 'Y'             | Sí, esta columna sólo contiene series de datos numéricos. IMPORTANTE: Si esta columna sólo contiene series numéricas seguidas de blancos de cola, no es aconsejable especificar 'Y'. | 'N' |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                 | 'N'             | No, esta columna no está limitada a series de datos numéricos.                                                                                                                       |     |
| <p>Al establecer serie_numérica en 'Y' para una columna, se informa al optimizador que esta columna no contiene blancos que puedan interferir en la clasificación de los datos de la columna. Esta opción sirve de ayuda cuando el orden de clasificación de una fuente de datos es diferente de DB2. Las columnas marcadas con esta opción no se excluirán de la evaluación local (base de datos) por un orden de clasificación diferente.</p> |                 |                                                                                                                                                                                      |     |

Tabla 123. Opciones de columna y sus valores (continuación)

| Opción                     | Valores válidos                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | Valor por omisión |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| varchar_no_trailing_blanks | <p>Especifica si la fuente de datos utiliza la semántica de comparación de varchar no rellena con blancos. Para series de caracteres de longitud variable que no contienen blancos de cola, alguna semántica de comparación no rellena con espacios en blanco de DBMS devuelve los mismos resultados que la semántica de comparación de DB2. Si está seguro de que ninguna columna de la tabla/vista VARCHAR de la fuente de datos contiene blancos de cola, considere establecer esta opción de servidor en 'Y' para una fuente de datos. Esta opción se utiliza con frecuencia con las fuentes de datos Oracle**.</p> <p>Asegúrese de que considera todos los objetos que potencialmente pueden tener apodos (incluyendo las vistas).</p> <p>'Y' Esta fuente de datos tiene una semántica de comparación no rellena con blancos similar a DB2.</p> <p>'N' Esta fuente de datos no tiene la misma semántica de comparación no rellena con blancos que DB2.</p> | 'N'               |

## Opciones de correlación de funciones

La finalidad principal de las opciones de correlación de funciones es proporcionar información acerca del coste potencial de ejecutar una función de fuente de datos en la fuente de datos. Si el análisis de bajada determina que se puede llamar a cualquiera de las dos funciones de una correlación, la información de estadísticas proporcionada en la definición de la correlación ayuda al optimizador a comparar el coste estimado de la ejecución de la función de la fuente de datos con el coste estimado de ejecutar la función de DB2.

Tabla 124. Opciones de correlación de funciones y sus valores

| Opción          | Valores válidos                                                                                                                | Valor por omisión |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------|-------------------|
| disable         | Inhabilita una correlación de funciones por omisión. Los valores válidos son 'Y' y 'N'.                                        | 'N'               |
| initial_insts   | El número estimado de instrucciones procesadas la primera y la última vez que se ha invocado la función de la fuente de datos. | '0'               |
| initial_ios     | Número estimado de E/S realizadas la primera y la última vez que se ha invocado la función de la fuente de datos.              | '0'               |
| ios_per_argbyte | Número estimado de E/S expandidas por cada byte del conjunto de argumentos que se pasa a la función de la fuente de datos.     | '0'               |
| ios_per_invoc   | Número estimado de E/S por invocación de una función de fuente de datos.                                                       | '0'               |

Tabla 124. Opciones de correlación de funciones y sus valores (continuación)

| Opción            | Valores válidos                                                                                                                   | Valor por omisión |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------|-------------------|
| insts_per_argbyte | Número estimado de instrucciones procesadas por cada byte del conjunto de argumentos que se pasa a la función de fuente de datos. | '0'               |
| insts_per_invoc   | Número estimado de instrucciones procesadas por invocación de la función de fuente de datos.                                      | '450'             |
| percent_argbytes  | Porcentaje promedio estimado de bytes de argumento de entrada que la función de fuente de datos leerá realmente.                  | '100'             |
| remote_name       | Nombre de la función de fuente de datos.                                                                                          | nombre local      |

### Opciones de servidor

Las opciones de servidor se utilizan para describir un servidor. Además de la información de ubicación (por ejemplo, el nombre de máquina de la fuente de datos), las opciones pueden especificar los atributos de seguridad y de rendimiento para una fuente de datos. Las opciones de seguridad proporcionan control de la comunicación con contraseña (enviada o no a las fuentes de datos) y de la autenticación de mayúsculas/minúsculas de la información (las mayúsculas y/o minúsculas de los ID y contraseñas). Las opciones de rendimiento ayudan al optimizador a determinar si las operaciones de evaluación se pueden realizar en las fuentes de datos y el mejor modelo de coste para completar las consultas que recuperan datos de las fuentes de datos.

Tabla 125. Opciones de servidor y sus valores

| Opción             | Valores válidos                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | Valor por omisión |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| collating_sequence | <p>Especifica si la fuente de datos utiliza el mismo orden de clasificación por omisión que la base de datos federada, basándose en el conjunto de códigos y la información del país. Si una fuente de datos tiene un orden de clasificación que difiere del orden de clasificación de DB2, la mayoría de operaciones que dependen del orden de clasificación de DB2 no se pueden evaluar remotamente en una fuente de datos. Un ejemplo es ejecutar funciones de columna MAX en una columna de caracteres de apodo en una fuente de datos con un orden de clasificación diferente. Puesto que los resultados pueden ser diferentes si la función MAX se evalúa en la fuente de datos remota, DB2 realizará la operación de agregación y la función MAX localmente.</p> <p>Si la consulta contiene un signo de igual, es posible bajar esa parte de la consulta incluso si los órdenes de clasificación son diferentes (establecida en 'N'). Por ejemplo, el predicado C1 = 'A' puede bajarse a una fuente de datos. Es evidente que tales consultas no se pueden bajar cuando el orden de clasificación de la fuente de datos es sensible a las mayúsculas y minúsculas. Cuando una fuente de datos no es sensible a las mayúsculas y minúsculas, los resultados de C1= 'A' y C1 = 'a' son iguales, lo que no es aceptable en un entorno sensible a las mayúsculas y minúsculas (DB2).</p> <p>Los administradores pueden crear bases de datos federadas con un orden de clasificación en particular que coincida con el orden de clasificación de la fuente de datos. Este enfoque puede acelerar el rendimiento si todas las fuentes de datos utilizan el mismo orden de clasificación o si la mayoría o todas las funciones de columna están dirigidas a las fuentes de datos que utilizan el mismo orden de clasificación.</p> <p>'Y' El orden de clasificación de la fuente de datos es igual al de la base de datos federada.</p> <p>'N' El orden de clasificación de la fuente de datos no es igual al de la base de datos federada.</p> <p>'I' El orden de clasificación de la fuente de datos es diferente al de la base de datos federada y no es sensible a las mayúsculas y minúsculas (por ejemplo, 'TOLLESON' y 'TolleSon' se consideran iguales).</p> | 'N'               |

Tabla 125. Opciones de servidor y sus valores (continuación)

| Opción        | Valores válidos                                                                                                                                                                                                                                                                                                                                                                                                                                       | Valor por omisión |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| comm_rate     | <p>Especifica la velocidad de comunicación entre un servidor federado y sus fuentes de datos asociadas. Expresada en megabytes por segundo.</p> <p>Los valores válidos son mayores que 0 y menores que 2147483648. Los valores sólo pueden ser números enteros, por ejemplo 12.</p>                                                                                                                                                                   | '2'               |
| connectstring | <p>Especifica las propiedades de inicialización necesarias para conectarse con un proveedor OLE DB. Para ver la sintaxis completa y la semántica de la serie de conexión, consulte el apartado "Data Link API of the OLE DB Core Components" de la publicación <i>Microsoft OLE DB 2.0 Programmer's Reference and Data Access SDK, Microsoft Press, 1998</i>.</p>                                                                                     | Ninguno           |
| cpu_ratio     | <p>Indica la diferencia de velocidad (en más o en menos) en que ejecuta la CPU de una fuente de datos con respecto a la CPU del servidor federado.</p> <p>Los valores válidos son mayores que 0 y menores que <math>1 \times 10^{23}</math>. Los valores pueden expresarse utilizando cualquier notación doble válida, por ejemplo 123E10, 123 ó 1.21E4.</p>                                                                                          | '1,0'             |
| dbname        | <p>Nombre de la base de datos de la fuente de datos a la que desea que el servidor federado acceda. Es necesario para las fuentes de datos de la familia DB2; no se aplica a las fuentes de datos Oracle**, pues las instancias de Oracle contienen una sola base de datos. Para DB2, este valor corresponde a una base de datos determinada contenida en una instancia o, en el caso de DB2 para OS/390, el valor LOCATION de una base de datos.</p> | Ninguno           |

Tabla 125. Opciones de servidor y sus valores (continuación)

| Opción                                                        | Valores válidos                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Valor por omisión |
|---------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| fold_id (Consulte las notas 1 y 4 al final de esta tabla.)    | <p>Se aplica a los ID de usuario que el servidor federado envía a las fuentes de datos para que los autentifiquen. Los valores válidos son:</p> <p>'U' El servidor federado convierte el ID de usuario a mayúsculas antes de enviarlo a la fuente de datos. Esta es una opción lógica para las fuentes de datos de la familia DB2 y Oracle** (Vea la nota 2 al final de esta tabla).</p> <p>'N' El servidor federado no realiza ninguna acción en el ID de usuario antes de enviarlo a la fuente de datos. (Consulte la nota 2 al final de esta tabla.)</p> <p>'L' El servidor federado convierte el ID de usuario a minúsculas antes de enviarlo a la fuente de datos.</p> <p>Si no se utiliza ninguno de estos valores, el servidor federado intenta enviar el ID de usuario a la fuente de datos en mayúsculas. Si el ID de usuario falla, el servidor intenta enviarlo en minúsculas.</p> | Ninguno           |
| fold_pw (Consulte las notas 1, 3 y 4 al final de esta tabla.) | <p>Se aplica a las contraseñas que el servidor federado envía a las fuentes de datos para que las autentifique. Los valores válidos son:</p> <p>'U' El servidor federado convierte la contraseña a mayúsculas antes de enviarla a la fuente de datos. Esta es una opción lógica para las fuentes de datos de la familia DB2 y Oracle**.</p> <p>'N' El servidor federado no realiza ninguna acción en la contraseña antes de enviarla a la fuente de datos.</p> <p>'L' El servidor federado convierte la contraseña a minúsculas antes de enviarla a la fuente de datos.</p> <p>Si no se utiliza ninguno de estos valores, el servidor federado intenta enviar la contraseña a la fuente de datos en mayúsculas. Si la contraseña falla, el servidor intenta enviarla en minúsculas.</p>                                                                                                       | Ninguno           |
| io_ratio                                                      | <p>Indica la diferencia de velocidad (en más o en menos) en que se ejecuta el sistema de E/S de la fuente de datos con respecto al sistema de E/S del servidor federado.</p> <p>Los valores válidos son mayores que 0 y menores que <math>1 \times 10^{23}</math>. Los valores pueden expresarse utilizando cualquier notación doble válida, por ejemplo 123E10, 123 ó 1.21E4.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | '1,0'             |



Tabla 125. Opciones de servidor y sus valores (continuación)

| Opción     | Valores válidos                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Valor por omisión |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| node       | <p>Nombre por el cual se define una fuente de datos como una instancia en su RDBMS. Es necesario para todas las fuentes de datos.</p> <p>Para una fuente de datos de la familia DB2, este nombre es el nodo especificado en el directorio de nodos DB2 de la base de datos federada. Para ver este directorio, emita el mandato <b>db2 list node directory</b>.</p> <p>Para una fuente de datos Oracle**, es el nombre de servidor especificado en el archivo Oracle** tnsnames.ora. Para acceder a este nombre en la plataforma Windows NT, especifique la opción <b>View Configuration Information</b> de la herramienta Oracle** SQL Net Easy Configuration.</p>                                                                   | Ninguno           |
| password   | <p>Especifica si se envían las contraseñas a una fuente de datos.</p> <p>'Y' Las contraseñas se envían siempre a la fuente de datos y se validan. Este es el valor por omisión.</p> <p>'N' Las contraseñas no se envían a la fuente de datos (sin tener en cuenta ninguna correlación de usuarios) y no se validan.</p> <p>'ENCRYPTION'<br/>Las contraseñas se envían siempre a la fuente de datos en forma cifrada y se validan. Sólo es válido para las fuentes de datos de la familia DB2 que soportan contraseñas cifradas.</p>                                                                                                                                                                                                   | 'Y'               |
| plan_hints | <p>Especifica si se han de habilitar las <i>indicaciones de planes</i>. Las indicaciones de planes son fragmentos de sentencias que proporcionan información adicional a los optimizadores de fuentes de datos. Esta información puede, para ciertos tipos de datos, mejorar el rendimiento de consultas. Las indicaciones de planes pueden ayudar al optimizador de fuentes de datos a decidir si se debe utilizar un índice, el índice que se ha de utilizar o qué orden de unión de tablas se ha de utilizar.</p> <p>'Y' Se han de habilitar las indicaciones de planes en la fuente de datos, si ésta soporta las indicaciones de planes.</p> <p>'N' No se han de habilitar las indicaciones de planes en la fuente de datos.</p> | 'N'               |

Tabla 125. Opciones de servidor y sus valores (continuación)

| Opción                     | Valores válidos                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Valor por omisión                                                                                                                                        |     |
|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| pushdown                   | 'Y'                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | DB2 tomará en consideración la posibilidad de dejar que la fuente de datos evalúe las operaciones.                                                       | 'Y' |
|                            | 'N'                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | DB2 sólo recuperará las columnas de la fuente de datos remota y no dejará que la fuente de datos evalúe las demás operaciones, por ejemplo, las uniones. |     |
| varchar_no_trailing_blanks | Especifica si la fuente de datos utiliza la semántica de comparación de varchar no rellena con blancos. Para series de caracteres de longitud variable que no contienen blancos de cola, alguna semántica de comparación no rellena con espacios en blanco de DBMS devuelve los mismos resultados que la semántica de comparación de DB2. Si está seguro de que ninguna columna de la tabla/vista VARCHAR de la fuente de datos contiene blancos de cola, considere establecer esta opción de servidor en 'Y' para una fuente de datos. Esta opción se utiliza con frecuencia con las fuentes de datos Oracle**. Asegúrese de que considera todos los objetos que potencialmente pueden tener apodos (incluyendo las vistas). |                                                                                                                                                          | 'N' |
|                            | 'Y'                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Esta fuente de datos tiene una semántica de comparación no rellena con blancos similar a DB2.                                                            |     |
|                            | 'N'                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Esta fuente de datos no tiene la misma semántica de comparación no rellena con blancos que DB2.                                                          |     |

Notas sobre la Tabla 125 en la página 1416:

1. Este campo se aplica sin tener en cuenta el valor especificado para autenticación.
2. Puesto que DB2 almacena los ID de usuario en mayúsculas, los valores 'N' y 'U' son equivalentes.
3. El valor de convertir\_ct no tiene ningún efecto cuando el valor de la contraseña es 'N'. Puesto que no se envía ninguna contraseña, las mayúsculas y minúsculas no pueden ser un factor.
4. Evite los valores nulos para estas opciones. Un valor nulo puede ser interesante porque DB2 intentará varias veces resolver los ID de usuario y contraseñas; sin embargo, puede afectar negativamente al rendimiento (es posible que DB2 envíe un ID de usuario y contraseña cuatro veces antes de pasar satisfactoriamente la autenticación de la fuente de datos).

### Opciones de usuario

Las opciones de usuario proporcionan información de autorización y de serie de contabilidad para las correlaciones de usuarios. Utilícelas para especificar

el ID y la contraseña utilizados para representar un ID de autenticación de DB2 cuando se autentifica en una fuente de datos.

Tabla 126. Opciones de usuario y sus valores

| Opción            | Valores válidos                                                                                                                                                                                                                                                                                                                            | Valor por omisión |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| remote_authid     | Indica el ID de autorización utilizado en la fuente de datos. Los valores válidos son cualquier serie de 255 de longitud o inferior. Si no se especifica esta opción, se utiliza el ID utilizado para conectarse a la base de datos.                                                                                                       | Ninguno           |
| dominio_remoto    | Indica el dominio de Windows NT que se utiliza para autenticar usuarios que se conectan a esta fuente de datos. Puede especificarse cualquier nombre de dominio válido de Windows NT. Si no se especifica esta opción, la fuente de datos realiza la autenticación utilizando el dominio de autenticación por omisión de la base de datos. | Ninguno           |
| remote_password   | Indica la contraseña de autorización utilizada en la fuente de datos. Puede especificarse cualquier serie de 32 caracteres de longitud o menos. Si no se especifica esta opción, se utiliza la contraseña utilizada para conectarse a la base de datos.                                                                                    | Ninguno           |
| accounting_string | Se utiliza para especificar una serie de contabilidad DRDA. Los valores válidos son cualquier serie de 255 de longitud o inferior. Esta opción sólo es necesaria si se necesita pasar información de contabilidad. Consulte el manual <i>DB2 Connect User's Guide</i>                                                                      | Ninguno           |

## Correlaciones de tipos de datos por omisión

Esta sección muestra las correlaciones por omisión entre los tipos de datos de DB2 soportados por el servidor federado y el tipo de datos soportado por las fuentes de datos siguientes:

- DB2 Universal Database para OS/390 y DB2 para MVS/ESA
- DB2 Universal Database para AS/400 y DB2 para OS/400
- Oracle
- DB2 para VM y VSE; SQL/DS

Las correlaciones mostradas son entre tipos de datos no idénticos. Las correlaciones entre tipos de datos idénticos no se muestran.

## Correlaciones de tipos por omisión entre fuentes de datos DB2 y DB2 Universal Database para OS/390 (y DB2 para MVS/ESA)

Tabla 127. Correlaciones de tipos por omisión entre fuentes de datos DB2 y DB2 Universal Database para OS/390 (y DB2 para MVS/ESA)

| DB2 para MVS, DB2 para OS/390 | DB2                             |
|-------------------------------|---------------------------------|
| varchar(n), n <= 32672        | varchar(n)                      |
| vargraphic(n), n <= 16336     | vargraphic(n)                   |
| char(255)                     | varchar(255)                    |
| char(255) para datos de bits  | varchar(255) para datos de bits |

## Correlaciones de tipos por omisión entre fuentes de datos DB2 y 2 Universal Database para AS/400 (y DB2 para OS/400)

Tabla 128. Correlaciones de tipos por omisión entre fuentes de datos DB2 y DB2 Universal Database para AS/400 (y DB2 para OS/400)

| DB2 para OS/400, DB2 para AS/400 | DB2           |
|----------------------------------|---------------|
| char(n), n <= 254                | char(n)       |
| char(n), n entre 255 y 32672     | varchar(n)    |
| varchar(n), n <= 32672           | varchar(n)    |
| graphic(n), n <= 127             | graphic(n)    |
| graphic(n), n entre 127 y 16336  | vargraphic(n) |
| vargraphic(n), n <= 16336        | vargraphic(n) |

## Correlaciones de tipos por omisión entre fuentes de datos DB2 y Oracle

Tabla 129. Correlaciones de tipos por omisión entre fuentes de datos DB2 y Oracle

| Oracle                             | DB2          |
|------------------------------------|--------------|
| rowid                              | char(18)     |
| char(n), n <= 254                  | char(n)      |
| nchar(n), n <= 254                 | char(n)      |
| char(255)                          | varchar(255) |
| varchar2(n), n <= 32672            | varchar(n)   |
| nvarchar2(n), n <= 32672           | varchar(n)   |
| number(p,s), p <= 4 y s = 0        | smallint     |
| number(p,s), 4 <= p <= 9 y s = 0   | entero       |
| number(p,s), 10 <= p <= 18 y s = 0 | bigint       |

Tabla 129. Correlaciones de tipos por omisión entre fuentes de datos DB2 y Oracle (continuación)

| Oracle                                                                               | DB2                             |
|--------------------------------------------------------------------------------------|---------------------------------|
| number(p,s), $p \leq 31$ y $0 \leq s \leq p$ y los dos casos anteriores no coinciden | decimal                         |
| number(p,s), todos los casos que no sean los 4 anteriores                            | double                          |
| raw(n), $n \leq 254$                                                                 | char(n) para datos de bits      |
| raw(255)                                                                             | varchar(255) para datos de bits |
| date (char(9))                                                                       | timestamp                       |

### Correlaciones de tipos por omisión entre fuentes de datos DB2 y DB2 para VM y VSE (y SQL/DS)

Tabla 130. Correlaciones de tipos por omisión entre fuentes de datos DB2 y DB2 para VM y VSE (y SQL/DS)

| DB2 para OS/390, SQL/DS       | DB2           |
|-------------------------------|---------------|
| varchar(n), $n \leq 32672$    | varchar(n)    |
| vargraphic(n), $n \leq 16336$ | vargraphic(n) |

### Proceso del recurso de paso a través

Se puede utilizar un recurso llamado *paso a través* para consultar una fuente de datos en el SQL nativo para esa fuente de datos. Esta sección:

- Indica las clases de sentencias de SQL de un servidor federado y sus fuentes de datos asociadas que se procesan en sesiones de paso a través.
- Lista las consideraciones y las restricciones que hay que tener en cuenta cuando se utiliza el paso a través.

### Proceso SQL en sesiones de paso a través

Las reglas siguientes especifican si una sentencia de SQL la procesa DB2 o una fuente de datos:

- Si se somete una sentencia de SQL para una fuente de datos a fin de que se procese en una sesión de paso a través, la sentencia se debe preparar dinámicamente en la sesión y ejecutarse mientras la sesión esté todavía abierta. Haya varias formas de hacer esto:
  - Si somete una sentencia SELECT, utilice la sentencia PREPARE para prepararla, luego utilice las sentencias OPEN, FETCH y CLOSE para acceder a los resultados de la consulta.
  - Para sentencias soportadas que no sean SELECT, siga uno de estos métodos:

- Utilice la sentencia PREPARE para preparar la sentencia soportada, y luego utilice la sentencia EXECUTE para ejecutarla.
- Utilice la sentencia EXECUTE IMMEDIATE para prepararla y ejecutarla.
- Si se somete una sentencia estática en una sesión de paso a través, se envía al servidor federado para su proceso.
- Si se emite un mandato COMMIT o ROLLBACK durante una sesión de paso a través, este mandato se completará en la unidad de trabajo (UOW) actual.

## Consideraciones y restricciones

Ciertas consideraciones y restricciones se aplican al paso a través. Algunas de ellas son de naturaleza general; otras sólo son relativas a las fuentes de datos Oracle.

### Utilización del paso a través con todas las fuentes de datos

La información siguiente se aplica a todas las fuentes de datos:

- Las sentencias preparadas dentro de una sesión de paso a través se deben ejecutar dentro de la misma sesión de paso a través. Las sentencias preparadas dentro de una sesión de paso a través, pero ejecutadas en una sesión diferente, no serán efectivas (SQLSTATE 56098).
- Los usuarios y aplicaciones pueden utilizar el paso a través para grabar en fuentes de datos; por ejemplo, para insertar, actualizar y suprimir filas de tabla. Observe que no se puede abrir un cursor directamente en un objeto de fuente de datos de una sesión de paso a través (SQLSTATE 25000).
- Una aplicación puede tener varias sentencias SET PASSTHRU en vigor a la vez para distintas fuentes de datos. Aunque la aplicación puede haber emitido varias sentencias SET PASSTHRU, las sesiones de paso a través no están verdaderamente anidadas. El servidor federado no realizará un paso a través de una fuente de datos para acceder a otra. En su lugar, el servidor accede a cada fuente directamente.
- Si hay varias sesiones de paso a través abiertas a la vez, cada unidad de trabajo de cada sesión debe concluirse con una sentencia COMMIT o ROLLBACK. Luego, se pueden finalizar las sesiones en una operación con la sentencia SET PASSTHRU y su opción RESET.
- No es posible realizar un paso a través para más de una fuente de datos a la vez.
- El paso a través no soporta las llamadas a procedimientos almacenados.
- La modalidad de paso a través no da soporte a la sentencia SELECT INTO.

### Utilización del paso a través con fuentes de datos Oracle

La siguiente información se aplica a las fuentes de datos Oracle:

- La siguiente restricción se aplica cuando un cliente remoto emite una sentencia SELECT desde el procesador de línea de mandatos (CLP) en la

modalidad de paso a través: Si el código del cliente es un SDK de DB2 anterior a DB2 Universal Database Versión 5, SELECT producirá un SQLSTATE 25000. Para evitar este error, los clientes remotos deben utilizar un SDK de DB2 que sea de la Versión 5 o posterior.

- Cualquier sentencia DDL emitida para un servidor Oracle se lleva a cabo en tiempo de análisis y no está sujeta a la semántica de las transacciones. Oracle confirma automáticamente la operación, una vez se ha completado. Si se produce una retroacción, el DDL no se retrotrae.
- Cuando se emite una sentencia SELECT desde tipos de datos sin formato, debe invocarse la función RAWTOHEX para recibir los valores hexadecimales. Cuando se realiza una operación INSERT en tipos de datos sin formato, debe proporcionarse la representación hexadecimal.





---

## Apéndice G. Tablas de la base de datos de ejemplo

Este apéndice muestra el contenido de las tablas de la base de datos de ejemplo SAMPLE y la manera de crearlas y eliminarlas.

Junto con DB2 Universal Database se proporcionan otras bases de datos de ejemplo para mostrar funciones sobre la información de la empresa y se utilizan en la guía de aprendizaje de esas funciones. Sin embargo, este apéndice sólo describe el contenido de la base de datos de ejemplo SAMPLE. Consulte la publicación *Data Warehouse Center Administration Guide* para obtener más información sobre las bases de datos de ejemplo referidas la información de la empresa.

Las tablas de muestra se utilizan en los ejemplos que aparecen en este manual y en otros manuales de esta biblioteca. Además, se muestran los datos contenidos en los archivos de muestra con tipos de datos BLOB y CLOB.

En este apéndice se incluyen las secciones siguientes:

- “La base de datos de muestra” en la página 1428
- “Para crear la base de datos SAMPLE” en la página 1428
- “Para borrar la base de datos de muestra” en la página 1428
- “Tabla CL\_SCHED” en la página 1429
- “Tabla DEPARTMENT” en la página 1429
- “Tabla EMPLOYEE” en la página 1429
- “Tabla EMP\_ACT” en la página 1433
- “Tabla EMP\_PHOTO” en la página 1435
- “Tabla EMP\_RESUME” en la página 1435
- “Tabla IN\_TRAY” en la página 1436
- “Tabla ORG” en la página 1436
- “Tabla PROJECT” en la página 1437
- “Tabla SALES” en la página 1438
- “Tabla STAFF” en la página 1439
- “Tabla STAFFG” en la página 1440
- “Archivos de muestra con tipos de datos BLOB y CLOB” en la página 1441
- “Foto de Quintana” en la página 1441
- “Currículum vitae de Quintana” en la página 1441
- “Foto de Nicholls” en la página 1442
- “Currículum vitae de Nicholls” en la página 1443
- “Foto de Adamson” en la página 1444
- “Currículum vitae de Adamson” en la página 1444
- “Foto de Walker” en la página 1445
- “Currículum vitae de Walker” en la página 1446

## Tablas de la base de datos de ejemplo

En las tablas de muestra, un guión (-) indica un valor nulo.

---

### La base de datos de muestra

Los ejemplos de este manual utilizan una base de datos de muestra. Para utilizar los ejemplos, debe crear la base de datos SAMPLE. Para utilizarla, debe estar instalado el gestor de bases de datos.

#### Para crear la base de datos SAMPLE

Un archivo ejecutable crea la base de datos de ejemplo.<sup>117</sup> Para crear una base de datos, debe tener autorización SYSADM.

- **Cuando se utilizan plataformas basadas en UNIX**

Si utiliza el indicador de mandatos del sistema operativo, escriba:

```
sql1lib/bin/db2samp1 <vía>
```

en el directorio inicial del propietario de la instancia del gestor de bases de datos, donde *vía de acceso* es un parámetro opcional que especifica la vía de acceso donde se ha de crear la base de datos de muestra. Pulse Intro.<sup>118</sup> El esquema para DB2SAMPL es el valor del registro especial CURRENT SCHEMA.

- **Cuando se utilizan plataformas OS/2 o Windows**

Si utiliza el indicador de mandatos del sistema operativo, escriba:

```
db2samp1 e
```

donde *e* es un parámetro opcional que especifica la unidad donde se ha de crear la base de datos. Pulse Intro.<sup>119</sup>

Si no se ha conectado a la estación de trabajo mediante Gestión de perfil de usuarios, se le solicitará que lo haga.

#### Para borrar la base de datos de muestra

Si no necesita acceder a la base de datos de muestra, puede borrarla utilizando el mandato DROP DATABASE:

```
db2 drop database sample
```

---

117. Para obtener información acerca de este mandato, consulte el mandato DB2SAMPL en el manual *Consulta de mandatos*.

118. Si no se especifica el parámetro de vía de acceso, las tablas de ejemplo se crea en la vía de acceso por omisión especificada por el parámetro DFTDBPATH en el archivo de configuración del gestor de bases de datos.

119. Si no se especifica el parámetro de unidad, la base de datos de ejemplo se crea en la misma unidad que DB2.

**Tabla CL\_SCHED**

| Nombre: | CLASS_CODE                   | DAY                        | STARTING         | ENDING         |
|---------|------------------------------|----------------------------|------------------|----------------|
| Tipo:   | char(7)                      | smallint                   | time             | time           |
| Desc:   | Class Code<br>(room:teacher) | Day # of 4 day<br>schedule | Class Start Time | Class End Time |

**Tabla DEPARTMENT**

| Nombre:  | DEPTNO               | DEPTNAME                                            | MGRNO                                                     | ADMRDEPT                                                         | LOCATION                       |
|----------|----------------------|-----------------------------------------------------|-----------------------------------------------------------|------------------------------------------------------------------|--------------------------------|
| Tipo:    | char(3) not null     | varchar(29) not null                                | char(6)                                                   | char(3) not null                                                 | char(16)                       |
| Desc:    | Department<br>number | Name describing general<br>activities of department | Employee<br>number<br>(EMPNO) of<br>department<br>manager | Department<br>(DEPTNO) to<br>which this<br>department<br>reports | Name of the<br>remote location |
| Valores: | A00                  | SPIFFY COMPUTER SERVICE<br>DIV.                     | 000010                                                    | A00                                                              | -                              |
|          | B01                  | PLANNING                                            | 000020                                                    | A00                                                              | -                              |
|          | C01                  | INFORMATION CENTER                                  | 000030                                                    | A00                                                              | -                              |
|          | D01                  | DEVELOPMENT CENTER                                  | -                                                         | A00                                                              | -                              |
|          | D11                  | MANUFACTURING SYSTEMS                               | 000060                                                    | D01                                                              | -                              |
|          | D21                  | ADMINISTRATION SYSTEMS                              | 000070                                                    | D01                                                              | -                              |
|          | E01                  | SUPPORT SERVICES                                    | 000050                                                    | A00                                                              | -                              |
|          | E11                  | OPERATIONS                                          | 000090                                                    | E01                                                              | -                              |
|          | E21                  | SOFTWARE SUPPORT                                    | 000100                                                    | E01                                                              | -                              |

**Tabla EMPLOYEE**

| Nombres: | EMPNO               | FIRSTNME                | MIDINIT             | LASTNAME                | WORKDEPT                                                    | PHONENO         | HIREDATE     |
|----------|---------------------|-------------------------|---------------------|-------------------------|-------------------------------------------------------------|-----------------|--------------|
| Tipo:    | char(6) not<br>null | varchar(12)<br>not null | char(1) not<br>null | varchar(15)<br>not null | char(3)                                                     | char(4)         | date         |
| Desc:    | Employee<br>number  | First name              | Middle<br>initial   | Last name               | Department<br>(DEPTNO)<br>in which the<br>employee<br>works | Phone<br>number | Date of hire |

| JOB     | EDLEVEL                                | SEX                          | BIRTHDATE     | SALARY        | BONUS        | COMM                 |
|---------|----------------------------------------|------------------------------|---------------|---------------|--------------|----------------------|
| char(8) | smallint not null                      | char(1)                      | date          | dec(9,2)      | dec(9,2)     | dec(9,2)             |
| Job     | Number of years of<br>formal education | Sex (M<br>male, F<br>female) | Date of birth | Yearly salary | Yearly bonus | Yearly<br>commission |

## Tablas de la base de datos de ejemplo

Consulte la página siguiente para ver los valores de la tabla EMPLOYEE.

## Tablas de la base de datos de ejemplo

| EMPNO               | FIRSTNAME               | MID INIT            | LASTNAME                | WORK DEPT | PHONE NO   | HIREDATE   | JOB      | ED LEVEL             | SEX     | BIRTHDATE  | SALARY   | BONUS    | COMM     |
|---------------------|-------------------------|---------------------|-------------------------|-----------|------------|------------|----------|----------------------|---------|------------|----------|----------|----------|
| char(6)<br>not null | varchar(12)<br>not null | char(1)<br>not null | varchar(15)<br>not null | char(3)   | char(4)    | date       | char(8)  | smallint<br>not null | char(1) | date       | dec(9,2) | dec(9,2) | dec(9,2) |
| 000010              | CHRISTINE               | I                   | HAAS                    | A00       | 3978       | 1965-01-01 | PRES     | 18                   | F       | 1933-08-24 | 52750    | 1000     | 4220     |
| 000020              | MICHAEL                 | L                   | THOMPSON                | B01       | 3476       | 1973-10-10 | MANAGER  | 18                   | M       | 1948-02-02 | 41250    | 800      | 3300     |
| 000030              | SALLY                   | A                   | KWAN                    | C01       | 4738       | 1975-04-05 | MANAGER  | 20                   | F       | 1941-05-11 | 38250    | 800      | 3060     |
| 000050              | JOHN                    | B                   | CHEYER                  | E01       | 6789       | 1949-08-17 | MANAGER  | 16                   | M       | 1925-09-15 | 40175    | 800      | 3214     |
| 000060              | IRVING                  | F                   | STERN                   | D11       | 6423       | 1973-09-14 | MANAGER  | 16                   | M       | 1945-07-07 | 32250    | 500      | 2580     |
| 000070              | EVA                     | D                   | PULASKI                 | D21       | 7831       | 1980-09-30 | MANAGER  | 16                   | F       | 1953-05-26 | 36170    | 700      | 2893     |
| 000090              | EILEEN                  | W                   | HENDERSON               | E11       | 5498       | 1970-08-15 | MANAGER  | 16                   | F       | 1941-05-15 | 29750    | 600      | 2380     |
| 000100              | THEODORE                | Q                   | SPENSER                 | E21       | 0972       | 1980-06-19 | MANAGER  | 14                   | M       | 1956-12-18 | 26150    | 500      | 2092     |
| 000110              | VINCENZO                | G                   | LUCCHESI                | A00       | 3490       | 1958-05-16 | SALESREP | 19                   | M       | 1929-11-05 | 46500    | 900      | 3720     |
| 000120              | SEAN                    | O'CONNELL           | A00                     | 2167      | 1963-12-05 | 1963-12-05 | CLERK    | 14                   | M       | 1942-10-18 | 29250    | 600      | 2340     |
| 000130              | DOLORES                 | M                   | QUINTANA                | C01       | 4578       | 1971-07-28 | ANALYST  | 16                   | F       | 1925-09-15 | 23800    | 500      | 1904     |
| 000140              | HEATHER                 | A                   | NICHOLLS                | C01       | 1793       | 1976-12-15 | ANALYST  | 18                   | F       | 1946-01-19 | 28420    | 600      | 2274     |
| 000150              | BRUCE                   | ADAMSON             | D11                     | 4510      | 1972-02-12 | 1972-02-12 | DESIGNER | 16                   | M       | 1947-05-17 | 25280    | 500      | 2022     |
| 000160              | ELIZABETH               | R                   | PIANKA                  | D11       | 3782       | 1977-10-11 | DESIGNER | 17                   | F       | 1955-04-12 | 22250    | 400      | 1780     |
| 000170              | MASATOSHI               | J                   | YOSHIMURA               | D11       | 2890       | 1978-09-15 | DESIGNER | 16                   | M       | 1951-01-05 | 24680    | 500      | 1974     |
| 000180              | MARILYN                 | S                   | SCOUTTEN                | D11       | 1682       | 1973-07-07 | DESIGNER | 17                   | F       | 1949-02-21 | 21340    | 500      | 1707     |
| 000190              | JAMES                   | H                   | WALKER                  | D11       | 2986       | 1974-07-26 | DESIGNER | 16                   | M       | 1952-06-25 | 20450    | 400      | 1636     |
| 000200              | DAVID                   | BROWN               | D11                     | 4501      | 1966-03-03 | 1966-03-03 | DESIGNER | 16                   | M       | 1941-05-29 | 27740    | 600      | 2217     |
| 000210              | WILLIAM                 | T                   | JONES                   | D11       | 0942       | 1979-04-11 | DESIGNER | 17                   | M       | 1953-02-23 | 18270    | 400      | 1462     |
| 000220              | JENNIFER                | K                   | LUTZ                    | D11       | 0672       | 1968-08-29 | DESIGNER | 18                   | F       | 1948-03-19 | 29840    | 600      | 2387     |
| 000230              | JAMES                   | J                   | JEFFERSON               | D21       | 2094       | 1966-11-21 | CLERK    | 14                   | M       | 1935-05-30 | 22180    | 400      | 1774     |
| 000240              | SALVATORE               | M                   | MARINO                  | D21       | 3780       | 1979-12-05 | CLERK    | 17                   | M       | 1954-03-31 | 28760    | 600      | 2301     |
| 000250              | DANIEL                  | S                   | SMITH                   | D21       | 0961       | 1969-10-30 | CLERK    | 15                   | M       | 1939-11-12 | 19180    | 400      | 1534     |
| 000260              | SYBIL                   | P                   | JOHNSON                 | D21       | 8953       | 1975-09-11 | CLERK    | 16                   | F       | 1936-10-05 | 17250    | 300      | 1380     |
| 000270              | MARIA                   | L                   | PEREZ                   | D21       | 9001       | 1980-09-30 | CLERK    | 15                   | F       | 1953-05-26 | 27380    | 500      | 2190     |
| 000280              | ETHEL                   | R                   | SCHNEIDER               | E11       | 8997       | 1967-03-24 | OPERATOR | 17                   | F       | 1936-03-28 | 26250    | 500      | 2100     |
| 000290              | JOHN                    | R                   | PARKER                  | E11       | 4502       | 1980-05-30 | OPERATOR | 12                   | M       | 1946-07-09 | 15340    | 300      | 1227     |
| 000300              | PHILIP                  | X                   | SMITH                   | E11       | 2095       | 1972-06-19 | OPERATOR | 14                   | M       | 1936-10-27 | 17750    | 400      | 1420     |
| 000310              | MAUDE                   | F                   | SETRIGHT                | E11       | 3332       | 1964-09-12 | OPERATOR | 12                   | F       | 1931-04-21 | 15900    | 300      | 1272     |
| 000320              | RAMLAL                  | V                   | MEHTA                   | E21       | 9990       | 1965-07-07 | FIELDREP | 16                   | M       | 1932-08-11 | 19950    | 400      | 1596     |

## Tablas de la base de datos de ejemplo

| EMPNO  | FIRSTNME | MID<br>INIT | LASTNAME | WORK<br>DEPT | PHONE<br>NO | HIREDATE   | JOB      | ED<br>LEVEL | SEX | BIRTHDATE  | SALARY | BONUS | COMM |
|--------|----------|-------------|----------|--------------|-------------|------------|----------|-------------|-----|------------|--------|-------|------|
| 000330 | WING     |             | LEE      | E21          | 2103        | 1976-02-23 | FIELDREP | 14          | M   | 1941-07-18 | 25370  | 500   | 2030 |
| 000340 | JASON    | R           | GOUNOT   | E21          | 5698        | 1947-05-05 | FIELDREP | 16          | M   | 1926-05-17 | 23840  | 500   | 1907 |

Tabla EMP\_ACT

| Nombre:  | EMPNO            | PROJNO           | ACTNO             | EMPTIME                                        | EMSTDATE             | EMENDATE           |
|----------|------------------|------------------|-------------------|------------------------------------------------|----------------------|--------------------|
| Tipo:    | char(6) not null | char(6) not null | smallint not null | dec(5,2)                                       | date                 | date               |
| Desc:    | Employee number  | Project number   | Activity number   | Proportion of employee's time spent on project | Date activity starts | Date activity ends |
| Valores: | 000010           | AD3100           | 10                | .50                                            | 1982-01-01           | 1982-07-01         |
|          | 000070           | AD3110           | 10                | 1.00                                           | 1982-01-01           | 1983-02-01         |
|          | 000230           | AD3111           | 60                | 1.00                                           | 1982-01-01           | 1982-03-15         |
|          | 000230           | AD3111           | 60                | .50                                            | 1982-03-15           | 1982-04-15         |
|          | 000230           | AD3111           | 70                | .50                                            | 1982-03-15           | 1982-10-15         |
|          | 000230           | AD3111           | 80                | .50                                            | 1982-04-15           | 1982-10-15         |
|          | 000230           | AD3111           | 180               | 1.00                                           | 1982-10-15           | 1983-01-01         |
|          | 000240           | AD3111           | 70                | 1.00                                           | 1982-02-15           | 1982-09-15         |
|          | 000240           | AD3111           | 80                | 1.00                                           | 1982-09-15           | 1983-01-01         |
|          | 000250           | AD3112           | 60                | 1.00                                           | 1982-01-01           | 1982-02-01         |
|          | 000250           | AD3112           | 60                | .50                                            | 1982-02-01           | 1982-03-15         |
|          | 000250           | AD3112           | 60                | .50                                            | 1982-12-01           | 1983-01-01         |
|          | 000250           | AD3112           | 60                | 1.00                                           | 1983-01-01           | 1983-02-01         |
|          | 000250           | AD3112           | 70                | .50                                            | 1982-02-01           | 1982-03-15         |
|          | 000250           | AD3112           | 70                | 1.00                                           | 1982-03-15           | 1982-08-15         |
|          | 000250           | AD3112           | 70                | .25                                            | 1982-08-15           | 1982-10-15         |
|          | 000250           | AD3112           | 80                | .25                                            | 1982-08-15           | 1982-10-15         |
|          | 000250           | AD3112           | 80                | .50                                            | 1982-10-15           | 1982-12-01         |
|          | 000250           | AD3112           | 180               | .50                                            | 1982-08-15           | 1983-01-01         |
|          | 000260           | AD3113           | 70                | .50                                            | 1982-06-15           | 1982-07-01         |
|          | 000260           | AD3113           | 70                | 1.00                                           | 1982-07-01           | 1983-02-01         |
|          | 000260           | AD3113           | 80                | 1.00                                           | 1982-01-01           | 1982-03-01         |
|          | 000260           | AD3113           | 80                | .50                                            | 1982-03-01           | 1982-04-15         |
|          | 000260           | AD3113           | 180               | .50                                            | 1982-03-01           | 1982-04-15         |
|          | 000260           | AD3113           | 180               | 1.00                                           | 1982-04-15           | 1982-06-01         |
|          | 000260           | AD3113           | 180               | .50                                            | 1982-06-01           | 1982-07-01         |
|          | 000270           | AD3113           | 60                | .50                                            | 1982-03-01           | 1982-04-01         |
|          | 000270           | AD3113           | 60                | 1.00                                           | 1982-04-01           | 1982-09-01         |
|          | 000270           | AD3113           | 60                | .25                                            | 1982-09-01           | 1982-10-15         |
|          | 000270           | AD3113           | 70                | .75                                            | 1982-09-01           | 1982-10-15         |
|          | 000270           | AD3113           | 70                | 1.00                                           | 1982-10-15           | 1983-02-01         |

## Tablas de la base de datos de ejemplo

| Nombre: | EMPNO  | PROJNO | ACTNO | EMPTIME | EMSTDATE   | EMENDATE   |
|---------|--------|--------|-------|---------|------------|------------|
|         | 000270 | AD3113 | 80    | 1.00    | 1982-01-01 | 1982-03-01 |
|         | 000270 | AD3113 | 80    | .50     | 1982-03-01 | 1982-04-01 |
|         | 000030 | IF1000 | 10    | .50     | 1982-06-01 | 1983-01-01 |
|         | 000130 | IF1000 | 90    | 1.00    | 1982-01-01 | 1982-10-01 |
|         | 000130 | IF1000 | 100   | .50     | 1982-10-01 | 1983-01-01 |
|         | 000140 | IF1000 | 90    | .50     | 1982-10-01 | 1983-01-01 |
|         | 000030 | IF2000 | 10    | .50     | 1982-01-01 | 1983-01-01 |
|         | 000140 | IF2000 | 100   | 1.00    | 1982-01-01 | 1982-03-01 |
|         | 000140 | IF2000 | 100   | .50     | 1982-03-01 | 1982-07-01 |
|         | 000140 | IF2000 | 110   | .50     | 1982-03-01 | 1982-07-01 |
|         | 000140 | IF2000 | 110   | .50     | 1982-10-01 | 1983-01-01 |
|         | 000010 | MA2100 | 10    | .50     | 1982-01-01 | 1982-11-01 |
|         | 000110 | MA2100 | 20    | 1.00    | 1982-01-01 | 1982-03-01 |
|         | 000010 | MA2110 | 10    | 1.00    | 1982-01-01 | 1983-02-01 |
|         | 000200 | MA2111 | 50    | 1.00    | 1982-01-01 | 1982-06-15 |
|         | 000200 | MA2111 | 60    | 1.00    | 1982-06-15 | 1983-02-01 |
|         | 000220 | MA2111 | 40    | 1.00    | 1982-01-01 | 1983-02-01 |
|         | 000150 | MA2112 | 60    | 1.00    | 1982-01-01 | 1982-07-15 |
|         | 000150 | MA2112 | 180   | 1.00    | 1982-07-15 | 1983-02-01 |
|         | 000170 | MA2112 | 60    | 1.00    | 1982-01-01 | 1983-06-01 |
|         | 000170 | MA2112 | 70    | 1.00    | 1982-06-01 | 1983-02-01 |
|         | 000190 | MA2112 | 70    | 1.00    | 1982-02-01 | 1982-10-01 |
|         | 000190 | MA2112 | 80    | 1.00    | 1982-10-01 | 1983-10-01 |
|         | 000160 | MA2113 | 60    | 1.00    | 1982-07-15 | 1983-02-01 |
|         | 000170 | MA2113 | 80    | 1.00    | 1982-01-01 | 1983-02-01 |
|         | 000180 | MA2113 | 70    | 1.00    | 1982-04-01 | 1982-06-15 |
|         | 000210 | MA2113 | 80    | .50     | 1982-10-01 | 1983-02-01 |
|         | 000210 | MA2113 | 180   | .50     | 1982-10-01 | 1983-02-01 |
|         | 000050 | OP1000 | 10    | .25     | 1982-01-01 | 1983-02-01 |
|         | 000090 | OP1010 | 10    | 1.00    | 1982-01-01 | 1983-02-01 |
|         | 000280 | OP1010 | 130   | 1.00    | 1982-01-01 | 1983-02-01 |
|         | 000290 | OP1010 | 130   | 1.00    | 1982-01-01 | 1983-02-01 |
|         | 000300 | OP1010 | 130   | 1.00    | 1982-01-01 | 1983-02-01 |
|         | 000310 | OP1010 | 130   | 1.00    | 1982-01-01 | 1983-02-01 |
|         | 000050 | OP2010 | 10    | .75     | 1982-01-01 | 1983-02-01 |
|         | 000100 | OP2010 | 10    | 1.00    | 1982-01-01 | 1983-02-01 |
|         | 000320 | OP2011 | 140   | .75     | 1982-01-01 | 1983-02-01 |



## Tablas de la base de datos de ejemplo

| Nombre: | EMPNO  | PROJNO | ACTNO | EMPTIME | EMSTDATE   | EMENDATE   |
|---------|--------|--------|-------|---------|------------|------------|
|         | 000320 | OP2011 | 150   | .25     | 1982-01-01 | 1983-02-01 |
|         | 000330 | OP2012 | 140   | .25     | 1982-01-01 | 1983-02-01 |
|         | 000330 | OP2012 | 160   | .75     | 1982-01-01 | 1983-02-01 |
|         | 000340 | OP2013 | 140   | .50     | 1982-01-01 | 1983-02-01 |
|         | 000340 | OP2013 | 170   | .50     | 1982-01-01 | 1983-02-01 |
|         | 000020 | PL2100 | 30    | 1.00    | 1982-01-01 | 1982-09-15 |

### Tabla EMP\_PHOTO

| Nombre:  | EMPNO            | PHOTO_FORMAT         | PICTURE           |
|----------|------------------|----------------------|-------------------|
| Tipo:    | char(6) not null | varchar(10) not null | blob(100k)        |
| Desc:    | Employee number  | Photo format         | Photo of employee |
| Valores: | 000130           | bitmap               | db200130.bmp      |
|          | 000130           | gif                  | db200130.gif      |
|          | 000130           | xwd                  | db200130.xwd      |
|          | 000140           | bitmap               | db200140.bmp      |
|          | 000140           | gif                  | db200140.gif      |
|          | 000140           | xwd                  | db200140.xwd      |
|          | 000150           | bitmap               | db200150.bmp      |
|          | 000150           | gif                  | db200150.gif      |
|          | 000150           | xwd                  | db200150.xwd      |
|          | 000190           | bitmap               | db200190.bmp      |
|          | 000190           | gif                  | db200190.gif      |
|          | 000190           | xwd                  | db200190.xwd      |

- “Foto de Quintana” en la página 1441 muestra la foto de la empleada, Dolores Quintana.
- “Foto de Nicholls” en la página 1442 muestra la foto de la empleada, Heather Nicholls.
- “Foto de Adamson” en la página 1444 muestra la foto del empleado, Bruce Adamson.
- “Foto de Walker” en la página 1445 muestra la foto del empleado, James Walker.

### Tabla EMP\_RESUME

| Nombre:  | EMPNO            | RESUME_FORMAT        | RESUME             |
|----------|------------------|----------------------|--------------------|
| Tipo:    | char(6) not null | varchar(10) not null | clob(5k)           |
| Desc:    | Employee number  | Resume Format        | Resume of employee |
| Valores: | 000130           | ascii                | db200130.asc       |

## Tablas de la base de datos de ejemplo

| Nombre: | EMPNO  | RESUME_FORMAT | RESUME       |
|---------|--------|---------------|--------------|
|         | 000130 | script        | db200130.scr |
|         | 000140 | ascii         | db200140.asc |
|         | 000140 | script        | db200140.scr |
|         | 000150 | ascii         | db200150.asc |
|         | 000150 | script        | db200150.scr |
|         | 000190 | ascii         | db200190.asc |
|         | 000190 | script        | db200190.scr |

- “Currículum vitae de Quintana” en la página 1441 muestra el currículum vitae de la empleada, Dolores Quintana.
- “Currículum vitae de Nicholls” en la página 1443 muestra el currículum vitae de la empleada, Heather Nicholls.
- “Currículum vitae de Adamson” en la página 1444 muestra el currículum vitae del empleado, Bruce Adamson.
- “Currículum vitae de Walker” en la página 1446 muestra el currículum vitae del empleado, James Walker.

### Tabla IN\_TRAY

| Nombre: | RECEIVED               | SOURCE                         | SUBJECT           | NOTE_TEXT     |
|---------|------------------------|--------------------------------|-------------------|---------------|
| Tipo:   | timestamp              | char(8)                        | char(64)          | varchar(3000) |
| Desc:   | Date and Time received | User id of person sending note | Brief description | The note      |

### Tabla ORG

| Nombre:  | DEPTNUMB          | DEPTNAME        | MANAGER        | DIVISION                | LOCATION      |
|----------|-------------------|-----------------|----------------|-------------------------|---------------|
| Tipo:    | smallint not null | varchar(14)     | smallint       | varchar(10)             | varchar(13)   |
| Desc:    | Department number | Department name | Manager number | Division of corporation | City          |
| Valores: | 10                | Head Office     | 160            | Corporate               | New York      |
|          | 15                | New England     | 50             | Eastern                 | Boston        |
|          | 20                | Mid Atlantic    | 10             | Eastern                 | Washington    |
|          | 38                | South Atlantic  | 30             | Eastern                 | Atlanta       |
|          | 42                | Great Lakes     | 100            | Midwest                 | Chicago       |
|          | 51                | Plains          | 140            | Midwest                 | Dallas        |
|          | 66                | Pacific         | 270            | Western                 | San Francisco |
|          | 84                | Mountain        | 290            | Western                 | Denver        |

Tabla PROJECT

| Nombre:  | PROJNO           | PROJNAME              | DEPTNO                 | RESPEMP              | PRSTAFF                 | PRSTDATE             | PRENDATE           | MAJPROJ                         |
|----------|------------------|-----------------------|------------------------|----------------------|-------------------------|----------------------|--------------------|---------------------------------|
| Tipo:    | char(6) not null | varchar(24) not null  | char(3) not null       | char(6) not null     | dec(5,2)                | date                 | date               | char(6)                         |
| Desc:    | Project number   | Project name          | Department responsible | Employee responsible | Estimated mean staffing | Estimated start date | Estimated end date | Major project, for a subproject |
| Valores: | AD3100           | ADMIN SERVICES        | D01                    | 000010               | 6.5                     | 1982-01-01           | 1983-02-01         | -                               |
|          | AD3110           | GENERAL ADMIN SYSTEMS | D21                    | 000070               | 6                       | 1982-01-01           | 1983-02-01         | AD3100                          |
|          | AD3111           | PAYROLL PROGRAMMING   | D21                    | 000230               | 2                       | 1982-01-01           | 1983-02-01         | AD3110                          |
|          | AD3112           | PERSONNEL PROGRAMMING | D21                    | 000250               | 1                       | 1982-01-01           | 1983-02-01         | AD3110                          |
|          | AD3113           | ACCOUNT PROGRAMMING   | D21                    | 000270               | 2                       | 1982-01-01           | 1983-02-01         | AD3110                          |
|          | IF1000           | QUERY SERVICES        | C01                    | 000030               | 2                       | 1982-01-01           | 1983-02-01         | -                               |
|          | IF2000           | USER EDUCATION        | C01                    | 000030               | 1                       | 1982-01-01           | 1983-02-01         | -                               |
|          | MA2100           | WELD LINE AUTOMATION  | D01                    | 000010               | 12                      | 1982-01-01           | 1983-02-01         | -                               |
|          | MA2110           | W L PROGRAMMING       | D11                    | 000060               | 9                       | 1982-01-01           | 1983-02-01         | MA2100                          |
|          | MA2111           | W L PROGRAM DESIGN    | D11                    | 000220               | 2                       | 1982-01-01           | 1982-12-01         | MA2110                          |
|          | MA2112           | W L ROBOT DESIGN      | D11                    | 000150               | 3                       | 1982-01-01           | 1982-12-01         | MA2110                          |
|          | MA2113           | W L PROD CONT PROGS   | D11                    | 000160               | 3                       | 1982-02-15           | 1982-12-01         | MA2110                          |
|          | OP1000           | OPERATION SUPPORT     | E01                    | 000050               | 6                       | 1982-01-01           | 1983-02-01         | -                               |
|          | OP1010           | OPERATION             | E11                    | 000090               | 5                       | 1982-01-01           | 1983-02-01         | OP1000                          |
|          | OP2000           | GEN SYSTEMS SERVICES  | E01                    | 000050               | 5                       | 1982-01-01           | 1983-02-01         | -                               |
|          | OP2010           | SYSTEMS SUPPORT       | E21                    | 000100               | 4                       | 1982-01-01           | 1983-02-01         | OP2000                          |
|          | OP2011           | SCP SYSTEMS SUPPORT   | E21                    | 000320               | 1                       | 1982-01-01           | 1983-02-01         | OP2010                          |
|          | OP2012           | APPLICATIONS SUPPORT  | E21                    | 000330               | 1                       | 1982-01-01           | 1983-02-01         | OP2010                          |
|          | OP2013           | DB/DC SUPPORT         | E21                    | 000340               | 1                       | 1982-01-01           | 1983-02-01         | OP2010                          |
|          | PL2100           | WELD LINE PLANNING    | B01                    | 000020               | 1                       | 1982-01-01           | 1982-09-15         | MA2100                          |

## Tablas de la base de datos de ejemplo

### Tabla SALES

| Nombre:  | SALES_DATE    | SALES_PERSON         | REGION          | SALES           |
|----------|---------------|----------------------|-----------------|-----------------|
| Tipo:    | date          | varchar(15)          | varchar(15)     | int             |
| Desc:    | Date of sales | Employee's last name | Region of sales | Number of sales |
| Valores: | 12/31/1995    | LUCCHESSI            | Ontario-South   | 1               |
|          | 12/31/1995    | LEE                  | Ontario-South   | 3               |
|          | 12/31/1995    | LEE                  | Quebec          | 1               |
|          | 12/31/1995    | LEE                  | Manitoba        | 2               |
|          | 12/31/1995    | GOUNOT               | Quebec          | 1               |
|          | 03/29/1996    | LUCCHESSI            | Ontario-South   | 3               |
|          | 03/29/1996    | LUCCHESSI            | Quebec          | 1               |
|          | 03/29/1996    | LEE                  | Ontario-South   | 2               |
|          | 03/29/1996    | LEE                  | Ontario-North   | 2               |
|          | 03/29/1996    | LEE                  | Quebec          | 3               |
|          | 03/29/1996    | LEE                  | Manitoba        | 5               |
|          | 03/29/1996    | GOUNOT               | Ontario-South   | 3               |
|          | 03/29/1996    | GOUNOT               | Quebec          | 1               |
|          | 03/29/1996    | GOUNOT               | Manitoba        | 7               |
|          | 03/30/1996    | LUCCHESSI            | Ontario-South   | 1               |
|          | 03/30/1996    | LUCCHESSI            | Quebec          | 2               |
|          | 03/30/1996    | LUCCHESSI            | Manitoba        | 1               |
|          | 03/30/1996    | LEE                  | Ontario-South   | 7               |
|          | 03/30/1996    | LEE                  | Ontario-North   | 3               |
|          | 03/30/1996    | LEE                  | Quebec          | 7               |
|          | 03/30/1996    | LEE                  | Manitoba        | 4               |
|          | 03/30/1996    | GOUNOT               | Ontario-South   | 2               |
|          | 03/30/1996    | GOUNOT               | Quebec          | 18              |
|          | 03/30/1996    | GOUNOT               | Manitoba        | 1               |
|          | 03/31/1996    | LUCCHESSI            | Manitoba        | 1               |
|          | 03/31/1996    | LEE                  | Ontario-South   | 14              |
|          | 03/31/1996    | LEE                  | Ontario-North   | 3               |
|          | 03/31/1996    | LEE                  | Quebec          | 7               |
|          | 03/31/1996    | LEE                  | Manitoba        | 3               |
|          | 03/31/1996    | GOUNOT               | Ontario-South   | 2               |
|          | 03/31/1996    | GOUNOT               | Quebec          | 1               |
|          | 04/01/1996    | LUCCHESSI            | Ontario-South   | 3               |
|          | 04/01/1996    | LUCCHESSI            | Manitoba        | 1               |
|          | 04/01/1996    | LEE                  | Ontario-South   | 8               |
|          | 04/01/1996    | LEE                  | Ontario-North   | -               |
|          | 04/01/1996    | LEE                  | Quebec          | 8               |
|          | 04/01/1996    | LEE                  | Manitoba        | 9               |
|          | 04/01/1996    | GOUNOT               | Ontario-South   | 3               |

## Tablas de la base de datos de ejemplo

| Nombre: | SALES_DATE | SALES_PERSON | REGION        | SALES |
|---------|------------|--------------|---------------|-------|
|         | 04/01/1996 | GOUNOT       | Ontario-North | 1     |
|         | 04/01/1996 | GOUNOT       | Quebec        | 3     |
|         | 04/01/1996 | GOUNOT       | Manitoba      | 7     |

### Tabla STAFF

| Nombre:  | ID                | NAME          | DEPT              | JOB      | YEARS            | SALARY         | COMM       |
|----------|-------------------|---------------|-------------------|----------|------------------|----------------|------------|
| Tipo:    | smallint not null | varchar(9)    | smallint          | char(5)  | smallint         | dec(7,2)       | dec(7,2)   |
| Desc:    | Employee number   | Employee name | Department number | Job type | Years of service | Current salary | Commission |
| Valores: | 10                | Sanders       | 20                | Mgr      | 7                | 18357.50       | -          |
|          | 20                | Pernal        | 20                | Sales    | 8                | 18171.25       | 612.45     |
|          | 30                | Marengi       | 38                | Mgr      | 5                | 17506.75       | -          |
|          | 40                | O'Brien       | 38                | Sales    | 6                | 18006.00       | 846.55     |
|          | 50                | Hanes         | 15                | Mgr      | 10               | 20659.80       | -          |
|          | 60                | Quigley       | 38                | Sales    | -                | 16808.30       | 650.25     |
|          | 70                | Rothman       | 15                | Sales    | 7                | 16502.83       | 1152.00    |
|          | 80                | James         | 20                | Clerk    | -                | 13504.60       | 128.20     |
|          | 90                | Koonitz       | 42                | Sales    | 6                | 18001.75       | 1386.70    |
|          | 100               | Plotz         | 42                | Mgr      | 7                | 18352.80       | -          |
|          | 110               | Ngan          | 15                | Clerk    | 5                | 12508.20       | 206.60     |
|          | 120               | Naughton      | 38                | Clerk    | -                | 12954.75       | 180.00     |
|          | 130               | Yamaguchi     | 42                | Clerk    | 6                | 10505.90       | 75.60      |
|          | 140               | Fraye         | 51                | Mgr      | 6                | 21150.00       | -          |
|          | 150               | Williams      | 51                | Sales    | 6                | 19456.50       | 637.65     |
|          | 160               | Molinare      | 10                | Mgr      | 7                | 22959.20       | -          |
|          | 170               | Kermisch      | 15                | Clerk    | 4                | 12258.50       | 110.10     |
|          | 180               | Abrahams      | 38                | Clerk    | 3                | 12009.75       | 236.50     |
|          | 190               | Sneider       | 20                | Clerk    | 8                | 14252.75       | 126.50     |
|          | 200               | Scoutten      | 42                | Clerk    | -                | 11508.60       | 84.20      |
|          | 210               | Lu            | 10                | Mgr      | 10               | 20010.00       | -          |
|          | 220               | Smith         | 51                | Sales    | 7                | 17654.50       | 992.80     |
|          | 230               | Lundquist     | 51                | Clerk    | 3                | 13369.80       | 189.65     |
|          | 240               | Daniels       | 10                | Mgr      | 5                | 19260.25       | -          |
|          | 250               | Wheeler       | 51                | Clerk    | 6                | 14460.00       | 513.30     |
|          | 260               | Jones         | 10                | Mgr      | 12               | 21234.00       | -          |
|          | 270               | Lea           | 66                | Mgr      | 9                | 18555.50       | -          |
|          | 280               | Wilson        | 66                | Sales    | 9                | 18674.50       | 811.50     |

## Tablas de la base de datos de ejemplo

| Nombre: | ID  | NAME     | DEPT | JOB   | YEARS | SALARY   | COMM    |
|---------|-----|----------|------|-------|-------|----------|---------|
|         | 290 | Quill    | 84   | Mgr   | 10    | 19818.00 | -       |
|         | 300 | Davis    | 84   | Sales | 5     | 15454.50 | 806.10  |
|         | 310 | Graham   | 66   | Sales | 13    | 21000.00 | 200.30  |
|         | 320 | Gonzales | 66   | Sales | 4     | 16858.20 | 844.00  |
|         | 330 | Burke    | 66   | Clerk | 1     | 10988.00 | 55.50   |
|         | 340 | Edwards  | 84   | Sales | 7     | 17844.00 | 1285.00 |
|         | 350 | Gafney   | 84   | Clerk | 5     | 13030.50 | 188.00  |

### Tabla STAFFG

**Nota:** STAFFG sólo se crea para páginas de códigos de doble byte.

| Nombre:  | ID                | NAME          | DEPT              | JOB        | YEARS            | SALARY         | COMM       |
|----------|-------------------|---------------|-------------------|------------|------------------|----------------|------------|
| Tipo:    | smallint not null | vargraphic(9) | smallint          | graphic(5) | smallint         | dec(9,0)       | dec(9,0)   |
| Desc:    | Employee number   | Employee name | Department number | Job type   | Years of service | Current salary | Commission |
| Valores: | 10                | Sanders       | 20                | Mgr        | 7                | 18357.50       | -          |
|          | 20                | Pernal        | 20                | Sales      | 8                | 18171.25       | 612.45     |
|          | 30                | Marenghi      | 38                | Mgr        | 5                | 17506.75       | -          |
|          | 40                | O'Brien       | 38                | Sales      | 6                | 18006.00       | 846.55     |
|          | 50                | Hanes         | 15                | Mgr        | 10               | 20659.80       | -          |
|          | 60                | Quigley       | 38                | Sales      | -                | 16808.30       | 650.25     |
|          | 70                | Rothman       | 15                | Sales      | 7                | 16502.83       | 1152.00    |
|          | 80                | James         | 20                | Clerk      | -                | 13504.60       | 128.20     |
|          | 90                | Koonitz       | 42                | Sales      | 6                | 18001.75       | 1386.70    |
|          | 100               | Plotz         | 42                | Mgr        | 7                | 18352.80       | -          |
|          | 110               | Ngan          | 15                | Clerk      | 5                | 12508.20       | 206.60     |
|          | 120               | Naughton      | 38                | Clerk      | -                | 12954.75       | 180.00     |
|          | 130               | Yamaguchi     | 42                | Clerk      | 6                | 10505.90       | 75.60      |
|          | 140               | Fraye         | 51                | Mgr        | 6                | 21150.00       | -          |
|          | 150               | Williams      | 51                | Sales      | 6                | 19456.50       | 637.65     |
|          | 160               | Molinare      | 10                | Mgr        | 7                | 22959.20       | -          |
|          | 170               | Kermisch      | 15                | Clerk      | 4                | 12258.50       | 110.10     |
|          | 180               | Abrahams      | 38                | Clerk      | 3                | 12009.75       | 236.50     |
|          | 190               | Sneider       | 20                | Clerk      | 8                | 14252.75       | 126.50     |
|          | 200               | Scoutten      | 42                | Clerk      | -                | 11508.60       | 84.20      |
|          | 210               | Lu            | 10                | Mgr        | 10               | 20010.00       | -          |
|          | 220               | Smith         | 51                | Sales      | 7                | 17654.50       | 992.80     |

## Tablas de la base de datos de ejemplo

| Nombre: | ID  | NAME      | DEPT | JOB   | YEARS | SALARY   | COMM    |
|---------|-----|-----------|------|-------|-------|----------|---------|
|         | 230 | Lundquist | 51   | Clerk | 3     | 13369.80 | 189.65  |
|         | 240 | Daniels   | 10   | Mgr   | 5     | 19260.25 | -       |
|         | 250 | Wheeler   | 51   | Clerk | 6     | 14460.00 | 513.30  |
|         | 260 | Jones     | 10   | Mgr   | 12    | 21234.00 | -       |
|         | 270 | Lea       | 66   | Mgr   | 9     | 18555.50 | -       |
|         | 280 | Wilson    | 66   | Sales | 9     | 18674.50 | 811.50  |
|         | 290 | Quill     | 84   | Mgr   | 10    | 19818.00 | -       |
|         | 300 | Davis     | 84   | Sales | 5     | 15454.50 | 806.10  |
|         | 310 | Graham    | 66   | Sales | 13    | 21000.00 | 200.30  |
|         | 320 | Gonzales  | 66   | Sales | 4     | 16858.20 | 844.00  |
|         | 330 | Burke     | 66   | Clerk | 1     | 10988.00 | 55.50   |
|         | 340 | Edwards   | 84   | Sales | 7     | 17844.00 | 1285.00 |
|         | 350 | Gafney    | 84   | Clerk | 5     | 13030.50 | 188.00  |

### Archivos de muestra con tipos de datos BLOB y CLOB

Esta sección muestra los datos encontrados en los archivos EMP\_PHOTO (fotos de empleados) y en los archivos EMP\_RESUME (resúmenes de empleados).

#### Foto de Quintana



*Figura 15. Dolores M. Quintana*

#### Currículum vitae de Quintana

El texto siguiente se encuentra en los archivos db200130.asc y db200130.scr.

#### Resume: Dolores M. Quintana

##### Personal Information

## Tablas de la base de datos de ejemplo

**Address:** 1150 Eglinton Ave Mellonville, Idaho 83725  
**Phone:** (208) 555-9933  
**Birthdate:** September 15, 1925  
**Sex:** Female  
**Marital Status:** Married  
**Height:** 5'2"  
**Weight:** 120 lbs.

### Department Information

**Employee Number:** 000130  
**Dept Number:** C01  
**Manager:** Sally Kwan  
**Position:** Analyst  
**Phone:** (208) 555-4578  
**Hire Date:** 1971-07-28

### Education

1965 Math and English, B.A. Adelphi University

1960 Dental Technician Florida Institute of Technology

### Work History

10/91 - present Advisory Systems Analyst Producing documentation tools for engineering department.

12/85 - 9/91 Technical Writer Writer, text programmer, and planner.

1/79 - 11/85 COBOL Payroll Programmer Writing payroll programs for a diesel fuel company.

### Interests

- Cooking
- Reading
- Sewing
- Remodeling

## Foto de Nicholls





Figura 16. Heather A. Nicholls

### **Currículum vitae de Nicholls**

El texto siguiente se encuentra en los archivos db200140.asc y db200140.scr.

#### **Resume: Heather A. Nicholls**

##### **Personal Information**

|                        |                                            |
|------------------------|--------------------------------------------|
| <b>Address:</b>        | 844 Don Mills Ave Mellonville, Idaho 83734 |
| <b>Phone:</b>          | (208) 555-2310                             |
| <b>Birthdate:</b>      | January 19, 1946                           |
| <b>Sex:</b>            | Female                                     |
| <b>Marital Status:</b> | Single                                     |
| <b>Height:</b>         | 5'8"                                       |
| <b>Weight:</b>         | 130 lbs.                                   |

##### **Department Information**

|                         |                |
|-------------------------|----------------|
| <b>Employee Number:</b> | 000140         |
| <b>Dept Number:</b>     | C01            |
| <b>Manager:</b>         | Sally Kwan     |
| <b>Position:</b>        | Analyst        |
| <b>Phone:</b>           | (208) 555-1793 |
| <b>Hire Date:</b>       | 1976-12-15     |

##### **Education**

|      |                                                      |
|------|------------------------------------------------------|
| 1972 | Computer Engineering, Ph.D. University of Washington |
| 1969 | Music and Physics, M.A. Vassar College               |

##### **Work History**

## Tablas de la base de datos de ejemplo

2/83 - present

Architect, OCR Development Designing the architecture of OCR products.

12/76 - 1/83

Text Programmer Optical character recognition (OCR) programming in PL/I.

9/72 - 11/76

Punch Card Quality Analyst Checking punch cards met quality specifications.

### Interests

- Model railroading
- Interior decorating
- Embroidery
- Knitting

### Foto de Adamson



Figura 17. Bruce Adamson

### Currículum vitae de Adamson

El texto siguiente se encuentra en los archivos db200150.asc y db200150.scr.

#### Resume: Bruce Adamson

##### Personal Information

|                        |                                           |
|------------------------|-------------------------------------------|
| <b>Address:</b>        | 3600 Steeles Ave Mellonville, Idaho 83757 |
| <b>Phone:</b>          | (208) 555-4489                            |
| <b>Birthdate:</b>      | May 17, 1947                              |
| <b>Sex:</b>            | Male                                      |
| <b>Marital Status:</b> | Married                                   |
| <b>Height:</b>         | 6'0"                                      |
| <b>Weight:</b>         | 175 lbs.                                  |

##### Department Information

## Tablas de la base de datos de ejemplo

**Employee Number:** 000150  
**Dept Number:** D11  
**Manager:** Irving Stern  
**Position:** Designer  
**Phone:** (208) 555-4510  
**Hire Date:** 1972-02-12

### Education

1971 Environmental Engineering, M.Sc. Johns Hopkins University

1968 American History, B.A. Northwestern University

### Work History

8/79 - present Neural Network Design Developing neural networks for machine intelligence products.

2/72 - 7/79 Robot Vision Development Developing rule-based systems to emulate sight.

9/71 - 1/72 Numerical Integration Specialist Helping bank systems communicate with each other.

### Interests

- Racing motorcycles
- Building loudspeakers
- Assembling personal computers
- Sketching

### Foto de Walker

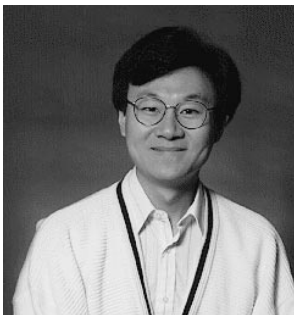


Figura 18. James H. Walker

## Tablas de la base de datos de ejemplo

### Currículum vitae de Walker

El texto siguiente se encuentra en los archivos db200190.asc y db200190.scr.

**Resume: James H. Walker**

#### Personal Information

**Address:** 3500 Steeles Ave Mellonville, Idaho 83757  
**Phone:** (208) 555-7325  
**Birthdate:** June 25, 1952  
**Sex:** Male  
**Marital Status:** Single  
**Height:** 5'11"  
**Weight:** 166 lbs.

#### Department Information

**Employee Number:** 000190  
**Dept Number:** D11  
**Manager:** Irving Stern  
**Position:** Designer  
**Phone:** (208) 555-2986  
**Hire Date:** 1974-07-26

#### Education

1974 Computer Studies, B.Sc. University of Massachusetts  
1972 Linguistic Anthropology, B.A. University of Toronto

#### Work History

6/87 - present Microcode Design Optimizing algorithms for mathematical functions.  
4/77 - 5/87 Printer Technical Support Installing and supporting laser printers.  
9/74 - 3/77 Maintenance Programming Patching assembly language compiler for mainframes.

#### Interests

- Wine tasting
- Skiing
- Swimming
- Dancing

---

## Apéndice H. Nombres de esquema reservados y palabras reservadas

Este apéndice describe las restricciones de algunos nombres utilizados por el gestor de bases de datos. En algunos casos, los nombres están reservados y los programas de aplicación no pueden utilizarlos. En otros casos, no es aconsejable la utilización de algunos nombres por programas de aplicación aunque no estén prohibidos por el gestor de bases de datos.

---

### Esquemas reservados

Los nombres de esquema siguientes están reservados:

- SYSCAT
- SYSFUN
- SYSIBM
- SYSSTAT

Además, se recomienda encarecidamente que los nombres de esquema no empiecen nunca por el prefijo SYS, ya que SYS se utiliza por convenio para indicar un área reservada por el sistema.

Las funciones no definidas por el usuario, los tipos definidos por el usuario, los desencadenantes o los seudónimos se pueden colocar en un esquema cuyo nombre empiece por SYS (SQLSTATE 42939).

También es recomendable no utilizar SESSION como nombre de esquema. Las tablas temporales declaradas deben estar calificadas por SESSION, por lo que es posible que una aplicación declare una tabla temporal que tenga el mismo nombre que el de una tabla permanente, lo cual puede producir confusión en la lógica del programa. Para evitar esta situación, no utilice el esquema SESSION excepto cuando utilice tablas temporales declaradas.

---

### Palabras reservadas

No hay palabras que estén específicamente reservadas en DB2 Versión 7.

Las palabras clave se pueden utilizar como identificadores normales, excepto en un contexto en que también se interpretarían como palabras clave SQL. En dichos casos, la palabra debe especificarse como un identificador delimitado. Por ejemplo, COUNT no se puede utilizar como un nombre de columna en una sentencia SELECT a menos que esté delimitada.

## **Nombres de esquema reservados y palabras reservadas**

SQL de IBM e ISO/ANSI SQL92 incluyen palabras reservadas, listadas en la siguiente sección. Estas palabras reservadas no se imponen por DB2 Universal Database; sin embargo, se recomienda que no se utilicen como identificadores normales, ya que reduce la portabilidad.

### Palabras reservadas de SQL de IBM

Las palabras reservadas de SQL de IBM son las siguientes.

|               |                   |            |              |
|---------------|-------------------|------------|--------------|
| ACQUIRE       | CONNECT           | EDITPROC   | IN           |
| ADD           | CONNECTION        | ELSE       | INDEX        |
| AFTER         | CONSTRAINT        | ELSEIF     | INDICATOR    |
| ALIAS         | CONTAINS          | END        | INNER        |
| ALL           | CONTINUE          | END-EXEC   | INOUT        |
| ALLOCATE      | COUNT             | ERASE      | INSENSITIVE  |
| ALLOW         | COUNT_BIG         | ESCAPE     | INSERT       |
| ALTER         | CREATE            | EXCEPT     | INTEGRITY    |
| AND           | CROSS             | EXCEPTION  | INTERSECT    |
| ANY           | CURRENT           | EXCLUSIVE  | INTO         |
| AS            | CURRENT_DATE      | EXECUTE    | IS           |
| ASC           | CURRENT_LC_PATH   | EXISTS     | ISOBID       |
| ASUTIME       | CURRENT_PATH      | EXIT       | ISOLATION    |
| AUDIT         | CURRENT_SERVER    | EXPLAIN    |              |
| AUTHORIZATION | CURRENT_TIME      | EXTERNAL   | JAVA         |
| AUX           | CURRENT_TIMESTAMP |            | JOIN         |
| AUXILIARY     | CURRENT_TIMEZONE  | FENCED     |              |
| AVG           | CURRENT_USER      | FETCH      | KEY          |
|               | CURSOR            | FIELDPROC  |              |
| BEFORE        |                   | FILE       | LABEL        |
| BEGIN         | DATA              | FINAL      | LANGUAGE     |
| BETWEEN       | DATABASE          | FOR        | LC_CTYPE     |
| BINARY        | DATE              | FOREIGN    | LEAVE        |
| BUFFERPOOL    | DAY               | FREE       | LEFT         |
| BY            | DAYS              | FROM       | LIKE         |
|               | DBA               | FULL       | LINKTYPE     |
| CALL          | DBINFO            | FUNCTION   | LOCAL        |
| CALLED        | DBSPACE           |            | LOCALE       |
| CAPTURE       | DB2GENERAL        | GENERAL    | LOCATOR      |
| CASCADE       | DB2SQL            | GENERATED  | LOCATORS     |
| CASE          | DECLARE           | GO         | LOCK         |
| CAST          | DEFAULT           | GOTO       | LOCKSIZE     |
| CCSID         | DELETE            | GRANT      | LONG         |
| CHAR          | DESC              | GRAPHIC    | LOOP         |
| CHARACTER     | DESCRIPTOR        | GROUP      |              |
| CHECK         | DETERMINISTIC     |            | MAX          |
| CLOSE         | DISALLOW          | HANDLER    | MICROSECOND  |
| CLUSTER       | DISCONNECT        | HAVING     | MICROSECONDS |
| COLLECTION    | DISTINCT          | HOURL      | MIN          |
| COLLID        | DO                | HOURS      | MINUTE       |
| COLUMN        | DOUBLE            |            | MINUTES      |
| COMMENT       | DROP              | IDENTIFIED | MODE         |
| COMMIT        | DSSIZE            | IF         | MODIFIES     |
| CONCAT        | DYNAMIC           | IMMEDIATE  | MONTH        |
| CONDITION     |                   |            | MONTHS       |

## Nombres de esquema reservados y palabras reservadas

|              |            |             |           |
|--------------|------------|-------------|-----------|
| NAME         | PACKAGE    | SCHEDULE    | UNDO      |
| NAMED        | PAGE       | SCHEMA      | UNION     |
| NHEADER      | PAGES      | SCRATCHPAD  | UNIQUE    |
| NO           | PARAMETER  | SECOND      | UNTIL     |
| NODENAME     | PART       | SECONDS     | UPDATE    |
| NODENUMBER   | PARTITION  | SECQTY      | USAGE     |
| NOT          | PATH       | SECURITY    | USER      |
| NULL         | PCTFREE    | SELECT      | USING     |
| NULLS        | PCTINDEX   | SET         |           |
| NUMPARTS     | PIECESIZE  | SHARE       | VALIDPROC |
|              | PLAN       | SIMPLE      | VALUES    |
| OBID         | POSITION   | SOME        | VARIABLE  |
| OF           | PRECISION  | SOURCE      | VARIANT   |
| ON           | PREPARE    | SPECIFIC    | VCAT      |
| ONLY         | PRIMARY    | SQL         | VIEW      |
| OPEN         | PRIQTY     | STANDARD    | VOLUMES   |
| OPTIMIZATION | PRIVATE    | STATIC      |           |
| OPTIMIZE     | PRIVILEGES | STATISTICS  | WHEN      |
| OPTION       | PROCEDURE  | STAY        | WHERE     |
| OR           | PROGRAM    | STOGROUP    | WHILE     |
| ORDER        | PSID       | STORES      | WITH      |
| OUT          | PUBLIC     | STORPOOL    | WLM       |
| OUTER        |            | STYLE       | WORK      |
|              | QUERYNO    | SUBPAGES    | WRITE     |
|              | READ       | SUBSTRING   |           |
|              | READS      | SUM         | YEAR      |
|              | RECOVERY   | SYNONYM     | YEARS     |
|              | REFERENCES | TABLE       |           |
|              | RELEASE    | TABLESPACE  |           |
|              | RENAME     | THEN        |           |
|              | REPEAT     | TO          |           |
|              | RESET      | TRANSACTION |           |
|              | RESOURCE   | TRIGGER     |           |
|              | RESTRICT   | TRIM        |           |
|              | RESULT     | TYPE        |           |
|              | RETURN     |             |           |
|              | RETURNS    |             |           |
|              | REVOKE     |             |           |
|              | RIGHT      |             |           |
|              | ROLLBACK   |             |           |
|              | ROW        |             |           |
|              | ROWS       |             |           |
|              | RRN        |             |           |
|              | RUN        |             |           |



### Palabras reservadas de ISO/ANS SQL92

Las palabras reservadas de ISO/ANS SQL92 que no están también en la lista de SQL de IBM son las siguientes.

|                  |           |              |                 |
|------------------|-----------|--------------|-----------------|
| ABSOLUTE         | EXEC      | NAMES        | SCROLL          |
| ACTION           | EXTRACT   | NATIONAL     | SECTION         |
| ARE              |           | NATURAL      | SESSION         |
| ASSERTION        | FALSE     | NCHAR        | SESSION_USER    |
| AT               | FIRST     | NEXT         | SIZE            |
|                  | FLOAT     | NULLIF       | SMALLINT        |
| BIT_LENGTH       | FOUND     | NUMERIC      | SPACE           |
| BOTH             | FULL      |              | SQLCODE         |
|                  |           | OCTET_LENGTH | SQLERROR        |
| CATALOG          | GET       | OUTPUT       | SQLSTATE        |
| CHAR_LENGTH      | GLOBAL    | OVERLAPS     | SYSTEM_USER     |
| CHARACTER_LENGTH |           |              |                 |
| COALESCE         | IDENTITY  | PAD          | TEMPORARY       |
| COLLATE          | INITIALLY | PARTIAL      | TIMEZONE_HOUR   |
| COLLATION        | INPUT     | PRESERVE     | TIMEZONE_MINUTE |
| CONSTRAINTS      | INTERVAL  | PRIOR        | TRAILING        |
| CONVERT          |           |              | TRANSLATION     |
| CORRESPONDING    | LAST      | REAL         | TRUE            |
|                  | LEADING   | RELATIVE     |                 |
| DEALLOCATE       | LEVEL     |              | UNKNOWN         |
| DEC              | LOWER     |              | UPPER           |
| DECIMAL          |           |              |                 |
| DEFERRABLE       | MATCH     |              | VALUE           |
| DEFERRED         | MODULE    |              | VARCHAR         |
| DESCRIBE         |           |              | VARYING         |
| DIAGNOSTICS      |           |              |                 |
| DOMAIN           |           |              | WHENEVER        |
|                  |           |              | ZONE            |

## Nombres de esquema reservados y palabras reservadas

## Apéndice I. Comparación de niveles de aislamiento

La tabla siguiente resume la información acerca de los niveles de aislamiento descritos en el apartado “Nivel de aislamiento” en la página 23.

|                                                                                                                                                        | UR                        | CS                        | RS | RR |
|--------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------|---------------------------|----|----|
| ¿La aplicación puede ver los cambios no confirmados realizados por otros procesos de aplicación?                                                       | Sí                        | No                        | No | No |
| ¿La aplicación puede actualizar los cambios no confirmados realizados por otros procesos de aplicación?                                                | No                        | No                        | No | No |
| ¿Pueden otros procesos de aplicación afectar a la operación de volver a ejecutar una sentencia? <i>Consulte el fenómeno P3 (fantasma) más abajo.</i>   | Sí                        | Sí                        | Sí | No |
| ¿Pueden otros procesos de aplicación actualizar filas “actualizadas”?                                                                                  | No                        | No                        | No | No |
| ¿Pueden otros procesos de aplicación que se ejecutan a un nivel de aislamiento que no sea UR leer las filas “actualizadas”?                            | No                        | No                        | No | No |
| ¿Pueden otros procesos de aplicación que se ejecutan al nivel de aislamiento UR leer las filas “actualizadas”?                                         | Sí                        | Sí                        | Sí | Sí |
| ¿Pueden otros procesos de aplicación actualizar las filas “a las que se ha accedido”? <i>Consulte el fenómeno P2 (lectura no repetible) más abajo.</i> | Sí                        | Sí                        | No | No |
| ¿Pueden otros procesos de aplicación leer las filas “a las que se ha accedido”?                                                                        | Sí                        | Sí                        | Sí | Sí |
| ¿Pueden otros procesos de aplicación actualizar o suprimir la fila “actual”? <i>Consulte el fenómeno P1 (lectura sucia) más abajo.</i>                 | Consulte la nota de abajo | Consulte la nota de abajo | No | No |

### Nota:

1. Si el cursor no es actualizable, con CS la fila actual puede actualizarse o suprimirse por otros procesos de aplicación en algunos casos.

## Niveles de aislamiento

UR CS RS RR

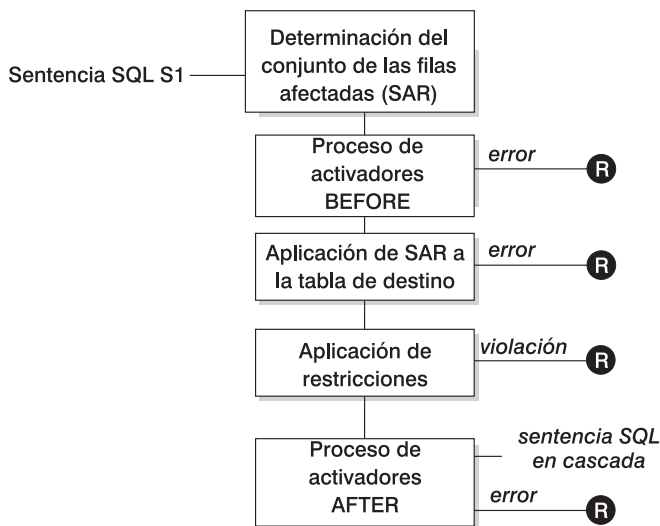
---

### Ejemplos de fenómenos:

- P1** *Lectura sucia.* La unidad de trabajo UW1 modifica una fila. La unidad de trabajo UW2 lee dicha fila antes que UW1 ejecute COMMIT. Si UW1 realiza entonces una operación ROLLBACK, UW2 ha leído una fila no existente.
- P2** *Lectura no repetible.* La unidad de trabajo UW1 lee una fila. La unidad de trabajo UW2 modifica dicha fila y realiza una operación COMMIT. Si UW1 vuelve a leer la fila, puede recibir un valor modificado.
- P3** *Fantasma.* La unidad de trabajo UW1 lee el conjunto de  $n$  filas que satisface alguna condición de búsqueda. La unidad de trabajo UW2 inserta (INSERT) una o varias filas que satisfacen la condición de búsqueda. Si UW1 repite la lectura inicial con la misma condición de búsqueda, obtiene las filas originales más las filas insertadas.

## Apéndice J. Interacción de desencadenantes y restricciones

Este apéndice describe la interacción de los desencadenantes con las restricciones de referencia y las restricciones de comprobación que pueden ser el resultado de una operación de actualización. La Figura 19 y la descripción asociada son representativas del proceso que se lleva a cabo para una sentencia de SQL que actualiza los datos de la base de datos.



**(R)** = retrotraer cambios antes de S1

Figura 19. Proceso de una sentencia de SQL con desencadenantes y restricciones asociados

La Figura 19 muestra el orden general de proceso para una sentencia de SQL que actualiza una tabla. Supone una situación donde la tabla incluye desencadenantes anteriores, restricciones de referencia, restricciones de comprobación y desencadenantes posteriores en cascada. A continuación encontrará una descripción de los recuadros y de los demás elementos que se encuentran en la Figura 19.

- Sentencia de SQL  $S_1$

Es la sentencia DELETE, INSERT o UPDATE que empieza el proceso. La sentencia de SQL  $S_1$  identifica una tabla (o una vista actualizable de alguna tabla) a la que se hace referencia como la *tabla de destino* en toda la descripción.

## Interacción de desencadenantes y restricciones

- Determinación del conjunto de las filas afectadas (*SAR*)

Este paso es el punto de arranque de un proceso que se repite para las reglas de supresión de restricción de referencia de CASCADE y SET NULL y para las sentencias de SQL en cascada de los desencadenantes posteriores.

La finalidad de este paso es determinar el *conjunto de filas afectadas* para la sentencia de SQL. El conjunto de filas incluidas en *SAR* se basa en la sentencia:

- para DELETE, todas las filas que satisfagan la condición de búsqueda de la sentencia (o la fila actual para una DELETE con posición)
- para INSERT, las filas identificadas por la cláusula VALUES o la selección completa
- para UPDATE, todas las filas que satisfagan la condición de búsqueda (o la fila actual para una actualización con posición)

Si *SAR* está vacío, no habrá desencadenantes BEFORE, los cambios sólo se aplican a la tabla de destino o a las restricciones para procesar la sentencia de SQL.

- Proceso de desencadenantes BEFORE

Todos los desencadenantes BEFORE se procesan por orden ascendente de creación. Cada desencadenante BEFORE procesará a acción activada una vez para cada fila de *SAR*.

Puede producirse un error durante el proceso de una acción activada en cuyo caso se retrotraen todos los cambios realizados como resultado de la sentencia de SQL original  $S_1$  (hasta entonces).

Si no hay ningún desencadenante BEFORE o si *SAR* está vacío, este paso se salta.

- Aplicación de *SAR* a la tabla de destino

La supresión, inserción o actualización real se aplica utilizando *SAR* en la tabla de destino de la base de datos.

Puede producirse un error al aplicar *SAR* (como el intento de insertar una fila con una clave duplicada donde existe un índice de unicidad) en cuyo caso se retrotraen todos los cambios realizados como resultado de la sentencia de SQL original  $S_1$  (hasta entonces).

- Aplicación de restricciones

Las restricciones asociadas con la tabla de destino se aplican si *SAR* no está vacío. Esto incluye restricciones de unicidad, índices de unicidad, restricciones de referencia, restricciones de comprobación y comprobaciones relacionadas con WITH CHECK OPTION en vistas. Las restricciones de referencia con reglas de supresión en cascada o de establecer nulo pueden provocar que se activen desencadenantes adicionales.

## Interacción de desencadenantes y restricciones

Una violación de cualquier restricción o WITH CHECK OPTION da como resultado un error y se retrotraen todos los cambios realizados como resultado de  $S_1$  (hasta entonces).

- Proceso de desencadenantes AFTER

Todos los desencadenantes AFTER activados por  $S_1$  se procesan por orden ascendente de creación.

Los desencadenantes FOR EACH procesan la acción activada una vez, incluso si SAR está vacío. Los desencadenantes FOR EACH ROW procesarán la acción activada una vez para cada fila de SAR.

Puede producirse un error durante el proceso de una acción activada, en cuyo caso se retrotraen todos los cambios realizados como resultado de la  $S_1$  original (hasta entonces).

La acción activada de un desencadenante puede incluir sentencias de SQL activadas que sean sentencias DELETE, INSERT o UPDATE. Para esta descripción, cada una de dichas sentencias se considera una *sentencia de SQL en cascada*.

Una sentencia de SQL en cascada es una sentencia DELETE, INSERT o UPDATE que se procesa como parte de la acción activada de un desencadenante AFTER. Esta sentencia empieza un nivel en cascada del proceso de desencadenante. Puede considerarse como la asignación de la sentencia de SQL activada como una  $S_1$  nueva y la realización repetida de todos los pasos descritos aquí.

Una vez que todas las sentencias de SQL activadas de todos los desencadenantes AFTER activados por cada  $S_1$  hayan terminado su proceso, el proceso de la  $S_1$  original se ha completado.

- **R** = retrotraer cambios antes de  $S_1$

Cualquier error que se produzca (incluyendo las violaciones de restricciones) durante el proceso da como resultado una retrotracción de todos los cambios realizados directa o indirectamente como resultado de la sentencia de SQL original  $S_1$ . Por lo tanto, la base de datos vuelve a estar en el mismo estado que estaba inmediatamente antes de la ejecución de la sentencia de SQL original  $S_1$ .

## Interacción de desencadenantes y restricciones



---

## Apéndice K. Tablas de Explain y definiciones

Las tablas de Explain capturan planes de acceso cuando se activa el recurso Explain. En esta sección se describen las siguientes tablas de Explain y definiciones:

- “Tabla EXPLAIN\_ARGUMENT” en la página 1460
- “Tabla EXPLAIN\_INSTANCE” en la página 1464
- “Tabla EXPLAIN\_OBJECT” en la página 1467
- “Tabla EXPLAIN\_OPERATOR” en la página 1469
- “Tabla EXPLAIN\_PREDICATE” en la página 1472
- “Tabla EXPLAIN\_STATEMENT” en la página 1474
- “Tabla EXPLAIN\_STREAM” en la página 1477
- “Tabla ADVISE\_INDEX” en la página 1479
- “Tabla ADVISE\_WORKLOAD” en la página 1482

Las tablas de Explain se deben crear antes de invocar Explain. Para crearlas, utilice el script de entrada del procesador de línea de mandatos de muestra, proporcionado en el archivo EXPLAIN.DDL ubicado en el subdirectorio 'misc' del directorio 'sqlib'. Conéctese a la base de datos donde se necesitan las tablas de Explain. Emita el mandato: `db2 -tf EXPLAIN.DDL` y se crearán las tablas. Vea “Definiciones de tabla para tablas de Explain” en la página 1483 para obtener más información.

Cuando el recurso Explain llene las tablas Explain no activará ningún desencadenante ni activará ninguna restricción de referencia ni de control. Por ejemplo, si se ha definido un desencadenante de inserción en la tabla EXPLAIN\_INSTANCE y se ha explicado una sentencia elegible, el desencadenante no se activará.

Para obtener más detalles sobre el recurso Explicar, consulte el manual *Administration Guide*.

### Leyenda para las tablas de Explain:

| <b>Cabecera</b>        | <b>Explain</b>                                                      |
|------------------------|---------------------------------------------------------------------|
| Nombre de columna      | El nombre de la columna                                             |
| Tipo de datos          | El tipo de datos de la columna                                      |
| ¿Posibilidad de nulos? | Sí: Están permitidos los nulos<br>No: Los nulos no están permitidos |

## Tablas de Explain

|             |                                                                                                     |
|-------------|-----------------------------------------------------------------------------------------------------|
| ¿Clave?     | PK: La columna forma parte de una clave primaria<br>FK: La columna forma parte de una clave foránea |
| Descripción | La descripción de la columna                                                                        |

---

### Tabla EXPLAIN\_ARGUMENT

La tabla EXPLAIN\_ARGUMENT representa las características exclusivas para cada operador individual, si hay alguno.

Para ver la definición de esta tabla, consulte el apartado “Definición de tabla EXPLAIN\_ARGUMENT” en la página 1484.

Tabla 131. Tabla EXPLAIN\_ARGUMENT

| Nombre de columna   | Tipo de datos | ¿Posibilidad de nulos? | ¿Clave? | Descripción                                                                                                                                                       |
|---------------------|---------------|------------------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EXPLAIN_REQUESTER   | VARCHAR(128)  | No                     | FK      | ID de autorización del iniciador de esta petición de Explain.                                                                                                     |
| EXPLAIN_TIME        | TIMESTAMP     | No                     | FK      | Hora de iniciación de la petición de Explain.                                                                                                                     |
| SOURCE_NAME         | VARCHAR(128)  | No                     | FK      | Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estática. |
| SOURCE_SCHEMA       | VARCHAR(128)  | No                     | FK      | Esquema, o calificador, de la fuente de la petición de Explain.                                                                                                   |
| EXPLAIN_LEVEL       | CHAR(1)       | No                     | FK      | Nivel de información de Explain para el que esta fila es aplicable.                                                                                               |
| STMTNO              | INTEGER       | No                     | FK      | Número de sentencia en el paquete con el que está relacionado esta información de Explain.                                                                        |
| SECTNO              | INTEGER       | No                     | FK      | Número de sección en el paquete con el que está relacionada esta información de Explain.                                                                          |
| OPERATOR_ID         | INTEGER       | No                     | No      | ID exclusivo para este operador en esta consulta.                                                                                                                 |
| ARGUMENT_TYPE       | CHAR(8)       | No                     | No      | El tipo de argumento para este operador.                                                                                                                          |
| ARGUMENT_VALUE      | VARCHAR(1024) | Sí                     | No      | El valor del argumento para este operador. NULL si el valor está en LONG_ARGUMENT_VALUE.                                                                          |
| LONG_ARGUMENT_VALUE | CLOB(1M)      | Sí                     | No      | El valor del argumento para este operador, cuando el texto no encaja ARGUMENT_VALUE. NULL si el valor está en ARGUMENT_VALUE.                                     |

Tabla 132. Valores de las columnas ARGUMENT\_TYPE y ARGUMENT\_VALUE

| Valor de ARGUMENT_TYPE | Valores de ARGUMENT_VALUE posibles                                                                                                                                                                                                                                                                                    | Descripción                                                                                |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|
| AGGMODE                | COMPLETE<br>PARTIAL<br>INTERMEDIATE<br>FINAL                                                                                                                                                                                                                                                                          | Indicadores de agregación parcial.                                                         |
| BITFLTR                | TRUE<br>FALSE                                                                                                                                                                                                                                                                                                         | La Unión de generación aleatoria utilizará un filtro de bits para mejorar el rendimiento.  |
| CSETEMP                | TRUE<br>FALSE                                                                                                                                                                                                                                                                                                         | Tabla temporal sobre el Distintivo de subexpresión común.                                  |
| DIRECT                 | TRUE                                                                                                                                                                                                                                                                                                                  | Indicador de lectura directa.                                                              |
| DUPLWARN               | TRUE<br>FALSE                                                                                                                                                                                                                                                                                                         | Duplica el distintivo de Aviso.                                                            |
| EARLYOUT               | TRUE<br>FALSE                                                                                                                                                                                                                                                                                                         | Indicador de pronto fuera.                                                                 |
| ENVVAR                 | Cada fila de este tipo contendrá: <ul style="list-style-type: none"> <li>• Nombre de la variable de entorno</li> <li>• Valor de la variable de entorno</li> </ul>                                                                                                                                                     | Variable de entorno que afecta al optimizador                                              |
| FETCHMAX               | IGNORE<br>INTEGER                                                                                                                                                                                                                                                                                                     | Altera temporalmente el valor del argumento MAXPAGES en el operador FETCH.                 |
| GROUPBYC               | TRUE<br>FALSE                                                                                                                                                                                                                                                                                                         | Si se proporcionan columnas Agrupar por.                                                   |
| GROUPBYN               | Integer                                                                                                                                                                                                                                                                                                               | Número de columnas de comparación.                                                         |
| GROUPBYR               | Cada fila de este tipo contendrá: <ul style="list-style-type: none"> <li>• El valor ordinal de la columna en grupo en clave (seguido por dos puntos y un espacio)</li> <li>• Nombre de columna</li> </ul>                                                                                                             | Requisito de Agrupar por.                                                                  |
| INNERCOL               | Cada fila de este tipo contendrá: <ul style="list-style-type: none"> <li>• El valor ordinal de la columna en orden (seguido por dos puntos y un espacio)</li> <li>• Nombre de columna</li> <li>• Valor de orden <ul style="list-style-type: none"> <li>(A) Ascendente</li> <li>(D) Descendente</li> </ul> </li> </ul> | Columnas de orden interno.                                                                 |
| ISCANMAX               | IGNORE<br>INTEGER                                                                                                                                                                                                                                                                                                     | Altera temporalmente el valor del argumento MAXPAGES en el operador ISCAN.                 |
| JN_INPUT               | INNER<br>OUTER                                                                                                                                                                                                                                                                                                        | Indica si el operador es el operador que alimenta la parte interna o externa de una unión. |
| LISTENER               | TRUE<br>FALSE                                                                                                                                                                                                                                                                                                         | Indicador de Cola de tabla receptora.                                                      |

## Tablas de Explain

Tabla 132. Valores de las columnas ARGUMENT\_TYPE y ARGUMENT\_VALUE (continuación)

| Valor de ARGUMENT_TYPE | Valores de ARGUMENT_VALUE posibles                                                                                                                                                                                                                                   | Descripción                                                                                                   |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------|
| MAXPAGES               | ALL<br>NONE<br>INTEGER                                                                                                                                                                                                                                               | Número máximo de páginas esperadas para la lectura anticipada.                                                |
| MAXRIDS                | NONE<br>INTEGER                                                                                                                                                                                                                                                      | Número máximo de Identificadores de fila que se incluirán en cada petición de lectura anticipada por lista.   |
| NUMROWS                | INTEGER                                                                                                                                                                                                                                                              | Número de filas que se espera clasificar.                                                                     |
| ONEFETCH               | TRUE<br>FALSE                                                                                                                                                                                                                                                        | Un indicador de lectura.                                                                                      |
| OUTERCOL               | Cada fila de este tipo contendrá: <ul style="list-style-type: none"> <li>• El valor ordinal de la columna en orden (seguido por dos puntos y un espacio)</li> <li>• Nombre de columna</li> <li>• Valor de orden</li> </ul> <p>(A) Ascendente<br/>(D) Descendente</p> | Columnas de orden externo.                                                                                    |
| OUTERJN                | LEFT<br>RIGHT                                                                                                                                                                                                                                                        | Indicador de unión externa.                                                                                   |
| PARTCOLS               | Nombre de columna                                                                                                                                                                                                                                                    | Columnas de partición para el operador.                                                                       |
| PREFETCH               | LIST<br>NONE<br>SEQUENTIAL                                                                                                                                                                                                                                           | Tipo de lectura anticipada admisible.                                                                         |
| RMTQTEXT               | Texto de consulta                                                                                                                                                                                                                                                    | Texto de consulta remoto                                                                                      |
| ROWLOCK                | EXCLUSIVE<br>NONE<br>REUSE<br>SHARE<br>SHORT (INSTANT) SHARE<br>UPDATE                                                                                                                                                                                               | Intento de bloqueo de fila.                                                                                   |
| ROWWIDTH               | INTEGER                                                                                                                                                                                                                                                              | Anchura de fila que se ha de clasificar.                                                                      |
| SCANDIR                | FORWARD<br>REVERSE                                                                                                                                                                                                                                                   | Dirección de exploración.                                                                                     |
| SCANGRAN               | INTEGER                                                                                                                                                                                                                                                              | Paralelismo intrapartición, granularidad de la exploración paralela de intrapartición, expresada en SCANUNIT. |
| SCANTYPE               | LOCAL PARALLEL                                                                                                                                                                                                                                                       | Paralelismo intrapartición, exploración de Índice o Tabla.                                                    |
| SCANUNIT               | ROW<br>PAGE                                                                                                                                                                                                                                                          | Paralelismo intrapartición, unidad de granularidad de exploración.                                            |
| SERVER                 | Servidor remoto                                                                                                                                                                                                                                                      | Servidor remoto                                                                                               |

Tabla 132. Valores de las columnas ARGUMENT\_TYPE y ARGUMENT\_VALUE (continuación)

| Valor de ARGUMENT_TYPE | Valores de ARGUMENT_VALUE posibles                                                                                                                                                                                                                                   | Descripción                                                                      |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------|
| SHARED                 | TRUE                                                                                                                                                                                                                                                                 | Paralelismo intrapartición, indicador TEMP compartido.                           |
| SLOWMAT                | TRUE<br>FALSE                                                                                                                                                                                                                                                        | Distintivo de materialización lenta.                                             |
| SNGLPROD               | TRUE<br>FALSE                                                                                                                                                                                                                                                        | Indicador SORT o TEMP de paralelismo intrapartición generado por un solo agente. |
| SORTKEY                | Cada fila de este tipo contendrá: <ul style="list-style-type: none"> <li>• El valor ordinal de la columna en clave (seguido por dos puntos y un espacio)</li> <li>• Nombre de columna</li> <li>• Valor de orden</li> </ul> <p>(A) Ascendente<br/>(D) Descendente</p> | Columnas de clave de clasificación.                                              |
| SORTTYPE               | PARTITIONED<br>SHARED<br>ROUND ROBIN<br>REPLICATED                                                                                                                                                                                                                   | Paralelismo intrapartición, tipo SORT.                                           |
| TABLOCK                | EXCLUSIVE<br>INTENT EXCLUSIVE<br>INTENT NONE<br>INTENT SHARE<br>REUSE<br>SHARE<br>SHARE INTENT EXCLUSIVE<br>SUPER EXCLUSIVE<br>UPDATE                                                                                                                                | Intento de bloqueo de tabla.                                                     |
| TQDEGREE               | INTEGER                                                                                                                                                                                                                                                              | paralelismo intrapartición, número de subagentes que acceden a la Cola de tabla. |
| TQMERGE                | TRUE<br>FALSE                                                                                                                                                                                                                                                        | Indicador de Fusión de cola de tabla (clasificada).                              |
| TQREAD                 | READ AHEAD<br>STEPPING<br>SUBQUERY STEPPING                                                                                                                                                                                                                          | Propiedad de lectura de Cola de tabla.                                           |
| TQSEND                 | BROADCAST<br>DIRECTED<br>SCATTER<br>SUBQUERY DIRECTED                                                                                                                                                                                                                | Propiedad de envío de Cola de tabla.                                             |
| TQTYPE                 | LOCAL                                                                                                                                                                                                                                                                | Paralelismo intrapartición, Cola de tabla.                                       |
| TRUNCSRT               | TRUE                                                                                                                                                                                                                                                                 | SORT truncado (limita el número de filas generadas).                             |
| UNIQUE                 | TRUE<br>FALSE                                                                                                                                                                                                                                                        | Indicador de exclusividad.                                                       |

## Tablas de Explain

Tabla 132. Valores de las columnas ARGUMENT\_TYPE y ARGUMENT\_VALUE (continuación)

| Valor de ARGUMENT_TYPE | Valores de ARGUMENT_VALUE posibles                                                                                                                                                            | Descripción                     |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------|
| UNIQUEKEY              | Cada fila de este tipo contendrá: <ul style="list-style-type: none"><li>• El valor ordinal de la columna en clave (seguido por dos puntos y un espacio)</li><li>• Nombre de columna</li></ul> | Columnas de claves de unicidad. |
| VOLATILE               | TRUE                                                                                                                                                                                          | Tabla volátil                   |

### Tabla EXPLAIN\_INSTANCE

La tabla EXPLAIN\_INSTANCE es la tabla principal de control para toda la información de Explain. Cada fila de datos de las tablas de Explain se enlaza explícitamente con una fila exclusiva de esta tabla. La tabla EXPLAIN\_INSTANCE proporciona la información básica acerca de la fuente de las sentencias de SQL que se están explicando, así como la información acerca del entorno en el que ha tenido lugar la explicación.

Para la definición de esta tabla, consulte el apartado “Definición de tabla EXPLAIN\_INSTANCE” en la página 1485.

Tabla 133. Tabla EXPLAIN\_INSTANCE

| Nombre de columna | Tipo de datos | ¿Posibilidad de nulos? | ¿Clave? | Descripción                                                                                                                                                       |
|-------------------|---------------|------------------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EXPLAIN_REQUESTER | VARCHAR(128)  | No                     | PK      | ID de autorización del iniciador de esta petición de Explain.                                                                                                     |
| EXPLAIN_TIME      | TIMESTAMP     | No                     | PK      | Hora de iniciación de la petición de Explain.                                                                                                                     |
| SOURCE_NAME       | VARCHAR(128)  | No                     | PK      | Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estática. |
| SOURCE_SCHEMA     | VARCHAR(128)  | No                     | PK      | Esquema, o calificador, de la fuente de la petición de Explain.                                                                                                   |
| EXPLAIN_OPTION    | CHAR(1)       | No                     | No      | Indica que se ha pedido la información de Explain para esta petición.                                                                                             |

Los posibles valores son:  
P PLAN SELECTION

Tabla 133. Tabla EXPLAIN\_INSTANCE (continuación)

| Nombre de columna | Tipo de datos | ¿Posibilidad de nulos? | ¿Clave? | Descripción                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-------------------|---------------|------------------------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SNAPSHOT_TAKEN    | CHAR(1)       | No                     | No      | Indica si se ha tomado una instantánea de Explain para esta petición.<br><br>Los posibles valores son:<br><b>Y</b> Sí, se ha(n) tomado una(s) instantánea(s) de Explain y se ha(n) almacenado en la tabla EXPLAIN_STATEMENT. También se ha capturado información de Explain normal.<br><b>N</b> No se ha tomado ninguna instantánea de Explain. Se ha capturado información de Explain normal.<br><b>O</b> Sólo se ha tomado una instantánea de Explain. No se ha capturado información de Explain normal. |
| DB2_VERSION       | CHAR(7)       | No                     | No      | Número de release del producto para DB2 Universal Database que ha procesado esta petición de Explain. El formato es vv.rr.m, donde:<br><b>vv</b> Número de versión<br><b>rr</b> Número de release<br><b>m</b> Número de release de mantenimiento                                                                                                                                                                                                                                                           |
| SQL_TYPE          | CHAR(1)       | No                     | No      | Indica si la instancia de Explain era para SQL dinámico o estático.<br><br>Los posibles valores son:<br><b>S</b> SQL estático<br><b>D</b> SQL dinámico                                                                                                                                                                                                                                                                                                                                                     |
| QUERYOPT          | INTEGER       | No                     | No      | Indica la clase de optimización de consulta que ha utilizado el Compilador SQL en el momento de la invocación de Explicar. El valor indica qué nivel de optimización de consulta ha efectuado el Compilador SQL para las sentencias de SQL que se explican.                                                                                                                                                                                                                                                |
| BLOCK             | CHAR(1)       | No                     | No      | Indica qué tipo de bloqueo de cursor se ha utilizado al compilar las sentencias de SQL. Para obtener información, consulte la columna BLOCK de SYSCAT.PACKAGES.<br><br>Los posibles valores son:<br><b>N</b> Sin bloqueo<br><b>U</b> Bloqueo de cursores no ambiguos<br><b>B</b> Bloqueo de todos los cursores                                                                                                                                                                                             |

## Tablas de Explain

Tabla 133. Tabla EXPLAIN\_INSTANCE (continuación)

| Nombre de columna | Tipo de datos | ¿Posibilidad de nulos? | ¿Clave? | Descripción                                                                                                                                                                                                                                                                                                                                  |
|-------------------|---------------|------------------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ISOLATION         | CHAR(2)       | No                     | No      | Indica qué tipo de aislamiento se ha utilizado al compilar las sentencias de SQL. Para obtener más información, consulte la columna ISOLATION en SYSCAT.PACKAGES.<br><br>Los posibles valores son:<br><b>RR</b> Lectura repetible<br><b>RS</b> Estabilidad de lectura<br><b>CS</b> Estabilidad del cursor<br><b>UR</b> Lectura no confirmada |
| BUFFPAGE          | INTEGER       | No                     | No      | Contiene el valor de configuración de base de datos BUFFPAGE en el momento de la invocación de Explicar.                                                                                                                                                                                                                                     |
| AVG_APPLS         | INTEGER       | No                     | No      | Contiene el valor del parámetro de configuración AVG_APPLS en el momento de la invocación de Explicar.                                                                                                                                                                                                                                       |
| SORTHEAP          | INTEGER       | No                     | No      | Contiene el valor de configuración de la base de datos SORTHEAP en el momento de la invocación de Explicar.                                                                                                                                                                                                                                  |
| LOCKLIST          | INTEGER       | No                     | No      | Contiene el valor de configuración de la base de datos LOCKLIST en el momento de la invocación de Explicar.                                                                                                                                                                                                                                  |
| MAXLOCKS          | SMALLINT      | No                     | No      | Contiene el valor de configuración de la base de datos MAXLOCKS en el momento de la invocación de Explicar.                                                                                                                                                                                                                                  |
| LOCKS_AVAIL       | INTEGER       | No                     | No      | Contiene el número de bloqueos que el optimizador supone que están disponibles para cada usuario. (Derivado de LOCKLIST y MAXLOCKS.)                                                                                                                                                                                                         |
| CPU_SPEED         | DOUBLE        | No                     | No      | Contiene el valor de configuración de la base de datos CPUSPEED en el momento de la invocación de Explicar.                                                                                                                                                                                                                                  |
| REMARKS           | VARCHAR(254)  | Sí                     | No      | Comentario proporcionado por el usuario.                                                                                                                                                                                                                                                                                                     |
| DBHEAP            | INTEGER       | No                     | No      | Contiene el valor de configuración de la base de datos DBHEAP en el momento de la invocación de Explicar.                                                                                                                                                                                                                                    |
| COMM_SPEED        | DOUBLE        | No                     | No      | Contiene el valor de configuración de la base de datos COMM_BANDWIDTH en el momento de la invocación de Explicar.                                                                                                                                                                                                                            |
| PARALLELISM       | CHAR(2)       | No                     | No      | Los posibles valores son: <ul style="list-style-type: none"> <li>• N = Sin paralelismo</li> <li>• P = Paralelismo intrapartición</li> <li>• IP = Paralelismo interparticiones</li> <li>• BP = Paralelismo intrapartición y paralelismo interparticiones</li> </ul>                                                                           |



Tabla 133. Tabla EXPLAIN\_INSTANCE (continuación)

| Nombre de columna | Tipo de datos | ¿Posibilidad de nulos? | ¿Clave? | Descripción                                                                                                                                               |
|-------------------|---------------|------------------------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| DATAJOINER        | CHAR(1)       | No                     | No      | Los posibles valores son: <ul style="list-style-type: none"> <li>• N = Plan de sistemas no federados</li> <li>• Y = Plan de sistemas federados</li> </ul> |

### Tabla EXPLAIN\_OBJECT

La tabla EXPLAIN\_OBJECT identifica los objetos de datos que necesita el plan de acceso generado para satisfacer la sentencia de SQL.

Para ver la definición de esta tabla, consulte el apartado “Definición de tabla EXPLAIN\_OBJECT” en la página 1486.

Tabla 134. Tabla EXPLAIN\_OBJECT

| Nombre de columna | Tipo de datos | ¿Posibilidad de nulos? | ¿Clave? | Descripción                                                                                                                                                       |
|-------------------|---------------|------------------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EXPLAIN_REQUESTER | VARCHAR(128)  | No                     | FK      | ID de autorización del iniciador de esta petición de Explain.                                                                                                     |
| EXPLAIN_TIME      | TIMESTAMP     | No                     | FK      | Hora de inicio de la petición de Explain.                                                                                                                         |
| SOURCE_NAME       | VARCHAR(128)  | No                     | FK      | Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estática. |
| SOURCE_SCHEMA     | VARCHAR(128)  | No                     | FK      | Esquema, o calificador, de la fuente de la petición de Explain.                                                                                                   |
| EXPLAIN_LEVEL     | CHAR(1)       | No                     | FK      | Nivel de información de Explain para el que esta fila es aplicable.                                                                                               |
| STMTNO            | INTEGER       | No                     | FK      | Número de sentencia en el paquete con el que está relacionado esta información de Explain.                                                                        |
| SECTNO            | INTEGER       | No                     | FK      | Número de sección en el paquete con el que está relacionada esta información de Explain.                                                                          |
| OBJECT_SCHEMA     | VARCHAR(128)  | No                     | No      | Esquema al que pertenece este objeto.                                                                                                                             |
| OBJECT_NAME       | VARCHAR(128)  | No                     | No      | Nombre del objeto.                                                                                                                                                |
| OBJECT_TYPE       | CHAR(2)       | No                     | No      | Etiqueta descriptiva del tipo de objeto.                                                                                                                          |
| CREATE_TIME       | TIMESTAMP     | Sí                     | No      | Hora de creación del objeto; nulo si es una función de tabla.                                                                                                     |
| STATISTICS_TIME   | TIMESTAMP     | Sí                     | No      | Última vez que se actualizaron las estadísticas para este objeto; nulo si no existen estadísticas para este objeto.                                               |
| COLUMN_COUNT      | SMALLINT      | No                     | No      | Número de columnas en este objeto.                                                                                                                                |
| ROW_COUNT         | INTEGER       | No                     | No      | Número estimado de filas en este objeto.                                                                                                                          |

## Tablas de Explain

Tabla 134. Tabla EXPLAIN\_OBJECT (continuación)

| Nombre de columna | Tipo de datos | ¿Posibilidad de nulos? | ¿Clave? | Descripción                                                                                                                                                                                                                                                                            |
|-------------------|---------------|------------------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| WIDTH             | INTEGER       | No                     | No      | La anchura promedio del objeto en bytes. Se establece en -1 para un índice.                                                                                                                                                                                                            |
| PAGES             | INTEGER       | No                     | No      | Número estimado de páginas que ocupa el objeto en la agrupación de almacenamientos intermedios. Establecido en -1 para una función de tabla.                                                                                                                                           |
| DISTINCT          | CHAR(1)       | No                     | No      | Indica si las filas del objeto son diferenciadas (p. ej., no hay duplicados)<br><br>Los posibles valores son:<br><b>Y</b> Sí<br><b>N</b> No                                                                                                                                            |
| TABLESPACE_NAME   | VARCHAR(128)  | Sí                     | No      | Nombre del espacio de tablas en el que está almacenado el objeto; se establece en nulo si no está implicado ningún espacio de tablas.                                                                                                                                                  |
| OVERHEAD          | DOUBLE        | No                     | No      | Actividad general total estimada, en milisegundos, para una sola E/S aleatoria para un espacio de tablas especificado. Incluye la actividad general del controlador, la búsqueda de disco y los tiempos de latencia. Se establece en -1 si no está implicado ningún espacio de tablas. |
| TRANSFER_RATE     | DOUBLE        | No                     | No      | Tiempo estimado para leer una página de datos, en milisegundos, del espacio de tablas especificado. Se establece en -1 si no está implicado ningún espacio de tablas.                                                                                                                  |
| PREFETCHSIZE      | INTEGER       | No                     | No      | Número de páginas de datos que se han de leer cuando se efectúa una lectura anticipada. Establecido en -1 para una función de tabla.                                                                                                                                                   |
| EXTENTSIZE        | INTEGER       | No                     | No      | El tamaño de extensión, en páginas de datos. Este volumen de páginas se escribe en un contenedor individual del espacio de tablas antes de cambiar al contenedor siguiente. Establecido en -1 para una función de tabla.                                                               |
| CLUSTER           | DOUBLE        | No                     | No      | Grado de agrupación de los datos con el índice. Si $\geq 1$ , es el CLUSTERRATIO. Si $\geq 0$ y $< 1$ , es el CLUSTERFACTOR. Se establece en -1 para una tabla, una función de tabla o si no está disponible esta estadística.                                                         |
| NLEAF             | INTEGER       | No                     | No      | Número de páginas que ocupan los valores de estos objetos de índice. Se establece en -1 para una tabla, una función de tabla o si no está disponible esta estadística.                                                                                                                 |
| NLEVELS           | INTEGER       | No                     | No      | Número de niveles de índice del árbol de este objeto de índice. Se establece en -1 para una tabla, una función de tabla o si no está disponible esta estadística.                                                                                                                      |

Tabla 134. Tabla EXPLAIN\_OBJECT (continuación)

| Nombre de columna | Tipo de datos | ¿Posibilidad de nulos? | ¿Clave? | Descripción                                                                                                                                                                                                                                                     |
|-------------------|---------------|------------------------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FULLKEYCARD       | BIGINT        | No                     | No      | Número de valores de clave completa diferenciada contenidos en este objeto de índice. Se establece en -1 para una tabla, una función de tabla o si no está disponible esta estadística.                                                                         |
| OVERFLOW          | INTEGER       | No                     | No      | Número total de registros de desbordamiento en la tabla. Se establece en -1 para un índice, una función de tabla o si no está disponible esta estadística.                                                                                                      |
| FIRSTKEYCARD      | BIGINT        | No                     | No      | Número de valores de primera clave diferenciada. Se establece en -1 para una tabla, una función de tabla o si no está disponible esta estadística.                                                                                                              |
| FIRST2KEYCARD     | BIGINT        | No                     | No      | Número de valores de primera clave diferenciada utilizando las primeras columnas {2,3,4} del índice. Se establece en -1 para una tabla, una función de tabla o si no está disponible esta estadística.                                                          |
| FIRST3KEYCARD     | BIGINT        | No                     | No      |                                                                                                                                                                                                                                                                 |
| FIRST4KEYCARD     | BIGINT        | No                     | No      |                                                                                                                                                                                                                                                                 |
| SEQUENTIAL_PAGES  | INTEGER       | No                     | No      | Número de páginas ubicadas en disco por orden de clave de índice con pocos o ningún vacío entre ellas. Se establece en -1 para una tabla, una función de tabla o si no está disponible esta estadística.                                                        |
| DENSITY           | INTEGER       | No                     | No      | Proporción de SEQUENTIAL_PAGES en relación al número de páginas del rango de páginas ocupado por el índice, expresada como porcentaje (entero entre 0 y 100). Se establece en -1 para una tabla, una función de tabla o si no está disponible esta estadística. |

Tabla 135. Valores de OBJECT\_TYPE posibles

| Valor | Descripción      |
|-------|------------------|
| IX    | Índice           |
| TA    | Tabla            |
| TF    | Función de tabla |

## Tabla EXPLAIN\_OPERATOR

La tabla EXPLAIN\_OPERATOR contiene todos los operadores necesarios para que el compilador SQL satisfaga la sentencia de SQL.

Para ver la definición de esta tabla, consulte el apartado “Definición de tabla EXPLAIN\_OPERATOR” en la página 1487.

## Tablas de Explain

Tabla 136. Tabla EXPLAIN\_OPERATOR

| Nombre de columna | Tipo de datos | ¿Posibilidad de nulos? | ¿Clave? | Descripción                                                                                                                                                                                     |
|-------------------|---------------|------------------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EXPLAIN_REQUESTER | VARCHAR(128)  | No                     | FK      | ID de autorización del iniciador de esta petición de Explain.                                                                                                                                   |
| EXPLAIN_TIME      | TIMESTAMP     | No                     | FK      | Hora de inicio de la petición de Explain.                                                                                                                                                       |
| SOURCE_NAME       | VARCHAR(128)  | No                     | FK      | Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estática.                               |
| SOURCE_SCHEMA     | VARCHAR(128)  | No                     | FK      | Esquema, o calificador, de la fuente de la petición de Explain.                                                                                                                                 |
| EXPLAIN_LEVEL     | CHAR(1)       | No                     | FK      | Nivel de información de Explain para el que esta fila es aplicable.                                                                                                                             |
| STMTNO            | INTEGER       | No                     | FK      | Número de sentencia en el paquete con el que está relacionado esta información de Explain.                                                                                                      |
| SECTNO            | INTEGER       | No                     | FK      | Número de sección en el paquete con el que está relacionada esta información de Explain.                                                                                                        |
| OPERATOR_ID       | INTEGER       | No                     | No      | ID exclusivo para este operador en esta consulta.                                                                                                                                               |
| OPERATOR_TYPE     | CHAR(6)       | No                     | No      | Etiqueta descriptiva para el tipo de operador.                                                                                                                                                  |
| TOTAL_COST        | DOUBLE        | No                     | No      | Coste total acumulado estimado (en timerons) de la ejecución del plan de acceso elegido hasta este operador inclusive.                                                                          |
| IO_COST           | DOUBLE        | No                     | No      | Coste de E/S acumulado estimado (en E/S de páginas de datos) de la ejecución del plan de acceso elegido hasta este operador (inclusive).                                                        |
| CPU_COST          | DOUBLE        | No                     | No      | Coste de CPU acumulado estimado (en las instrucciones) de la ejecución del plan de acceso elegido hasta este operador (inclusive).                                                              |
| FIRST_ROW_COST    | DOUBLE        | No                     | No      | Coste acumulado estimado (en timerons) de la lectura de la primera fila para el plan de acceso hasta este operador inclusive. Este valor incluye cualquier actividad general inicial necesaria. |
| RE_TOTAL_COST     | DOUBLE        | No                     | No      | Coste acumulado estimado (en timerons) de la lectura de la siguiente fila para el plan de acceso elegido hasta este operador inclusive.                                                         |
| RE_IO_COST        | DOUBLE        | No                     | No      | Coste de E/S acumulado estimado (en E/S de páginas de datos) de la lectura de la siguiente fila del plan de acceso elegido hasta este operador inclusive.                                       |
| RE_CPU_COST       | DOUBLE        | No                     | No      | Coste de CPU acumulado estimado (en timerons) de la lectura de la siguiente fila para el plan de acceso elegido hasta este operador inclusive.                                                  |
| COMM_COST         | DOUBLE        | No                     | No      | Coste de comunicación acumulado estimado (en tramas TCP/IP) de la ejecución del plan de acceso elegido hasta este operador (inclusive).                                                         |

Tabla 136. Tabla EXPLAIN\_OPERATOR (continuación)

| Nombre de columna | Tipo de datos | ¿Posibilidad de nulos? | ¿Clave? | Descripción                                                                                                                                                                                                             |
|-------------------|---------------|------------------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FIRST_COMM_COST   | DOUBLE        | No                     | No      | Coste de comunicaciones acumulado estimado (en TCP/IP) de la lectura de la primera fila para el plan de acceso elegido hasta este operador inclusive. Este valor incluye cualquier actividad general inicial necesaria. |
| BUFFERS           | DOUBLE        | No                     | No      | Requisitos estimados de almacenamiento intermedio para este operador y sus entradas.                                                                                                                                    |
| REMOTE_TOTAL_COST | DOUBLE        | No                     | No      | Coste total acumulado estimado (en timerons) de la ejecución de operación(es) en base(s) de datos remota(s).                                                                                                            |
| REMOTE_COMM_COST  | DOUBLE        | No                     | No      | Coste de comunicación acumulado estimado de la ejecución del plan de acceso remoto elegido hasta este operador (inclusive).                                                                                             |

Tabla 137. Valores de OPERATOR\_TYPE

| Valor  | Descripción                                     |
|--------|-------------------------------------------------|
| DELETE | Suprimir                                        |
| FETCH  | Leer                                            |
| FILTER | Filtrar filas                                   |
| GENROW | Generar fila                                    |
| GRPBY  | Agrupar por                                     |
| HSJOIN | Unión de generación aleatoria                   |
| INSERT | Insertar                                        |
| IXAND  | Aplicación de AND de Índice de bitmaps dinámico |
| IXSCAN | Exploración de índice                           |
| MSJOIN | Fusionar unión de exploración                   |
| NLJOIN | Unión de bucle anidado                          |
| RETURN | Resultado                                       |
| RIDSCN | Exploración de identificador de fila (RID)      |
| RQUERY | Consulta remota                                 |
| SORT   | Clasificar                                      |
| TBSCAN | Exploración de tabla                            |
| TEMP   | Construcción de tabla temporal                  |
| TQ     | Cola de tabla                                   |
| UNION  | Unión                                           |
| UNIQUE | Eliminación de duplicados                       |
| UPDATE | Actualizar                                      |

## Tablas de Explain

### Tabla EXPLAIN\_PREDICATE

La tabla EXPLAIN\_PREDICATE identifica los predicados que aplica un operador específico.

Para ver la definición de esta tabla, consulte el apartado “Definición de tabla EXPLAIN\_PREDICATE” en la página 1488.

Tabla 138. Tabla EXPLAIN\_PREDICATE

| Nombre de columna | Tipo de datos | ¿Posibilidad de nulos? | ¿Clave? | Descripción                                                                                                                                                       |
|-------------------|---------------|------------------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EXPLAIN_REQUESTER | VARCHAR(128)  | No                     | FK      | ID de autorización del iniciador de esta petición de Explain.                                                                                                     |
| EXPLAIN_TIME      | TIMESTAMP     | No                     | FK      | Hora de inicio de la petición de Explain.                                                                                                                         |
| SOURCE_NAME       | VARCHAR(128)  | No                     | FK      | Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estática. |
| SOURCE_SCHEMA     | VARCHAR(128)  | No                     | FK      | Esquema, o calificador, de la fuente de la petición de Explain.                                                                                                   |
| EXPLAIN_LEVEL     | CHAR(1)       | No                     | FK      | Nivel de información de Explain para el que esta fila es aplicable.                                                                                               |
| STMTNO            | INTEGER       | No                     | FK      | Número de sentencia en el paquete con el que está relacionado esta información de Explain.                                                                        |
| SECTNO            | INTEGER       | No                     | FK      | Número de sección en el paquete con el que está relacionada esta información de Explain.                                                                          |
| OPERATOR_ID       | INTEGER       | No                     | No      | ID exclusivo para este operador en esta consulta.                                                                                                                 |
| PREDICATE_ID      | INTEGER       | No                     | No      | ID exclusivo de este predicado para el operador especificado.                                                                                                     |
| HOW_APPLIED       | CHAR(5)       | No                     | No      | La forma en que el operador especificado utiliza el predicado.                                                                                                    |

Tabla 138. Tabla EXPLAIN\_PREDICATE (continuación)

| Nombre de columna | Tipo de datos | ¿Posibilidad de nulos? | ¿Clave? | Descripción                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-------------------|---------------|------------------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| WHEN_EVALUATED    | CHAR(3)       | No                     | No      | Indica cuándo se evalúa la subconsulta utilizada en este predicado.<br><br>Los posibles valores son:<br><br><b>blanco</b> Este predicado no contiene ninguna subconsulta.<br><br><b>EAA</b> La subconsulta utilizada en este predicado se evalúa en la aplicación (EAA). Es decir, se vuelve a evaluar para cada fila procesada por el operador especificado, cuando se aplica el predicado.<br><br><b>EAO</b> La subconsulta utilizada en este predicado se evalúa en la apertura (EAO). Es decir, se vuelve a evaluar sólo una vez para el operador especificado y sus resultados se vuelven a utilizar en la aplicación del predicado para cada fila.<br><br><b>MUL</b> Hay más de una subconsulta en este predicado. |
| RELOP_TYPE        | CHAR(2)       | No                     | No      | El tipo de operador relacional utilizado en este predicado.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| SUBQUERY          | CHAR(1)       | No                     | No      | Si es necesaria una corriente de datos de una subconsulta o no para este predicado. Puede ser necesarias múltiples corrientes de subconsultas.<br><br>Los posibles valores son:<br><br><b>N</b> No es necesaria ninguna corriente de subconsulta<br><br><b>Y</b> Son necesarias una o varias corrientes de subconsultas                                                                                                                                                                                                                                                                                                                                                                                                  |
| FILTER_FACTOR     | DOUBLE        | No                     | No      | La fracción estimada de filas que este predicado calificará.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| PREDICATE_TEXT    | CLOB(1M)      | Sí                     | No      | El texto del predicado tal como se ha vuelto a crear a partir de la representación interna de la sentencia de SQL.<br><br>Nulo si no está disponible.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

Tabla 139. Valores de HOW\_APPLIED posibles

| Valor | Descripción                         |
|-------|-------------------------------------|
| JOIN  | Utilizado para unir tablas          |
| RESID | Evaluado como un predicado residual |

## Tablas de Explain

Tabla 139. Valores de HOW\_APPLIED posibles (continuación)

| Valor | Descripción                                                            |
|-------|------------------------------------------------------------------------|
| SARG  | Evaluado como un predicado comparable para un índice o página de datos |
| START | Utilizado como una condición de inicio                                 |
| STOP  | Utilizado como una condición de detención                              |

Tabla 140. Valores de RELOP\_TYPE posibles

| Valor   | Descripción       |
|---------|-------------------|
| blancos | No aplicable      |
| EQ      | Igual             |
| GE      | Mayor o igual que |
| GT      | Mayor que         |
| IN      | En lista          |
| LE      | Menor o igual que |
| LK      | Igual             |
| LT      | Menor que         |
| NE      | Diferente a       |
| NL      | Es nulo           |
| NN      | No es nulo        |

## Tabla EXPLAIN\_STATEMENT

La tabla EXPLAIN\_STATEMENT contiene el texto de la sentencia de SQL tal como existe para los diferentes niveles de información de Explain. La sentencia de SQL original se almacena tal como la entra el usuario, en esta tabla junto con la versión utilizada (por el optimizador) para elegir el plan de acceso para satisfacer la sentencia de SQL. La última versión puede parecerse poco a la original ya que puede haberse vuelto a escribir y/o mejorar con predicados adicionales tal como lo determina el Compilador SQL.

Para la definición de esta tabla, consulte el apartado “Definición de tabla EXPLAIN\_STATEMENT” en la página 1489.

Tabla 141. Tabla EXPLAIN\_STATEMENT

| Nombre de columna | Tipo de datos | ¿Posibilidad de nulos? | ¿Clave? | Descripción                                                   |
|-------------------|---------------|------------------------|---------|---------------------------------------------------------------|
| EXPLAIN_REQUESTER | VARCHAR(128)  | No                     | PK, FK  | ID de autorización del iniciador de esta petición de Explain. |
| EXPLAIN_TIME      | TIMESTAMP     | No                     | PK, FK  | Hora de iniciación de la petición de Explain.                 |



Tabla 141. Tabla EXPLAIN\_STATEMENT (continuación)

| Nombre de columna | Tipo de datos | ¿Posibilidad de nullos? | ¿Clave? | Descripción                                                                                                                                                                                                                                                                                                                                                                                |
|-------------------|---------------|-------------------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SOURCE_NAME       | VARCHAR(128)  | No                      | PK, FK  | Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estática.                                                                                                                                                                                                                          |
| SOURCE_SCHEMA     | VARCHAR(128)  | No                      | PK, FK  | Esquema, o calificador, de la fuente de la petición de Explain.                                                                                                                                                                                                                                                                                                                            |
| EXPLAIN_LEVEL     | CHAR(1)       | No                      | PK      | Nivel de información de Explain para el que esta fila es aplicable.<br><br>Los valores válidos son:<br><b>O</b> Texto original (tal como lo ha entrado el usuario)<br><b>P</b> PLAN SELECTION                                                                                                                                                                                              |
| STMTNO            | INTEGER       | No                      | PK      | Número de sentencia en el paquete con el que está relacionado esta información de Explain. Establecido en 1 para las sentencias de SQL dinámicas de Explain. Para las sentencias de SQL estáticas, este valor es igual al valor utilizado para la vista de catálogo SYSCAT.STATEMENTS.                                                                                                     |
| SECTNO            | INTEGER       | No                      | PK      | Número de sección en el paquete que contiene esta sentencia de SQL. En las sentencias de SQL dinámicas de Explain, es el número de sección utilizada para contener la sección para esta sentencia en tiempo de ejecución. Para las sentencias de SQL estáticas, este valor es igual al valor utilizado para la vista de catálogo SYSCAT.STATEMENTS.                                        |
| QUERYNO           | INTEGER       | No                      | No      | Identificador numérico para la sentencia de SQL explicada. Para sentencias de SQL dinámicas (excluyendo la sentencia EXPLAIN SQL) emitidas a través de CLP o CLI, el valor por omisión es un valor incrementado secuencialmente. De lo contrario, el valor por omisión es el valor de STMTNO para sentencias de SQL estáticas y 1 para sentencias de SQL dinámicas.                        |
| QUERYTAG          | CHAR(20)      | No                      | No      | Distintivo identificador para cada sentencia de SQL explicada. Para sentencias de SQL dinámicas emitidas a través de CLP (excluyendo la sentencia EXPLAIN SQL), el valor por omisión es 'CLP'. Para sentencias de SQL dinámicas emitidas a través de CLI (excluyendo la sentencia EXPLAIN SQL), el valor por omisión es 'CLI'. De lo contrario, el valor por omisión utilizado es blancos. |

## Tablas de Explain

Tabla 141. Tabla EXPLAIN\_STATEMENT (continuación)

| Nombre de columna | Tipo de datos | ¿Posibilidad de nulos? | ¿Clave? | Descripción                                                                                                                                                                                                                                                                                                                             |
|-------------------|---------------|------------------------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| STATEMENT_TYPE    | CHAR(2)       | No                     | No      | Etiqueta descriptiva para el tipo de consulta que se está explicando.<br><br>Los posibles valores son:<br><b>S</b> Selección<br><b>D</b> Suprimir<br><b>DC</b> Supresión en la ubicación actual del cursor<br><b>I</b> Insertar<br><b>U</b> Actualizar<br><b>UC</b> Actualización en la ubicación actual del cursor                     |
| UPDATABLE         | CHAR(1)       | No                     | No      | Indica si esta sentencia se considera actualizable. Es relevante en particular para las sentencias SELECT que pueden determinarse como actualizables en potencia.<br><br>Los posibles valores son:<br><b>' '</b> No aplicable (blanco)<br><b>N</b> No<br><b>Y</b> Sí                                                                    |
| DELETABLE         | CHAR(1)       | No                     | No      | Indica si esta sentencia se considera suprimible. Es relevante en particular para las sentencias SELECT que pueden determinarse como suprimibles en potencia.<br><br>Los posibles valores son:<br><b>' '</b> No aplicable (blanco)<br><b>N</b> No<br><b>Y</b> Sí                                                                        |
| TOTAL_COST        | DOUBLE        | No                     | No      | Coste total estimado (en timerons) de la ejecución del plan de acceso elegido para esta sentencia; se establece en 0 (cero) si EXPLAIN_LEVEL es 0 (texto original) ya que no se ha elegido ningún plan de acceso en este momento.                                                                                                       |
| STATEMENT_TEXT    | CLOB(1M)      | No                     | No      | Texto o fragmento del texto de la sentencia de SQL que se está explicando. El texto que se muestra para el nivel de Explain de Selección del plan se ha reconstruido a partir de la representación interna y es parecido a SQL en su naturaleza; es decir, no se garantiza que la sentencia reconstruida siga la sintaxis SQL correcta. |

Tabla 141. Tabla EXPLAIN\_STATEMENT (continuación)

| Nombre de columna | Tipo de datos | ¿Posibilidad de nulos? | ¿Clave? | Descripción                                                                                                                                                                                                                                                                                                                                                                          |
|-------------------|---------------|------------------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SNAPSHOT          | BLOB(10M)     | Sí                     | No      | Instantánea de la representación interna de esta sentencia de SQL en el Nivel_Explain mostrado.<br><br>Esta columna está pensada para utilizarla con DB2 Visual Explain. La columna se establece en nulo si EXPLAIN_LEVEL es 0 (sentencia original), pues no se había elegido ningún plan de acceso todavía en el momento en que se capturó esta versión específica de la sentencia. |
| QUERY_DEGREE      | INTEGER       | No                     | No      | Indica el grado de paralelismo intrapartición en el momento de la invocación de Explicar. Para la sentencia original, contiene el grado dirigido de paralelismo intrapartición. Para PLAN SELECTION, contiene el grado de paralelismo intrapartición generado para que lo utilice el plan.                                                                                           |

### Tabla EXPLAIN\_STREAM

La tabla EXPLAIN\_STREAM representa las corrientes de datos de entrada y de salida entre operadores individuales y objetos de datos. Los objetos de datos en sí se representan en la tabla EXPLAIN\_OBJECT. Los operadores implicados en una corriente de datos se han de encontrar en una tabla EXPLAIN\_OPERATOR.

Para ver la definición de esta tabla, consulte el apartado “Definición de tabla EXPLAIN\_STREAM” en la página 1490.

Tabla 142. Tabla EXPLAIN\_STREAM

| Nombre de columna | Tipo de datos | ¿Posibilidad de nulos? | ¿Clave? | Descripción                                                                                                                                                       |
|-------------------|---------------|------------------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EXPLAIN_REQUESTER | VARCHAR(128)  | No                     | FK      | ID de autorización del iniciador de esta petición de Explain.                                                                                                     |
| EXPLAIN_TIME      | TIMESTAMP     | No                     | FK      | Hora de inicio de la petición de Explain.                                                                                                                         |
| SOURCE_NAME       | VARCHAR(128)  | No                     | FK      | Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estática. |
| SOURCE_SCHEMA     | VARCHAR(128)  | No                     | FK      | Esquema, o calificador, de la fuente de la petición de Explain.                                                                                                   |
| EXPLAIN_LEVEL     | CHAR(1)       | No                     | FK      | Nivel de información de Explain para el que esta fila es aplicable.                                                                                               |
| STMTNO            | INTEGER       | No                     | FK      | Número de sentencia en el paquete con el que está relacionado esta información de Explain.                                                                        |

## Tablas de Explain

Tabla 142. Tabla EXPLAIN\_STREAM (continuación)

| Nombre de columna | Tipo de datos | ¿Posibilidad de nulos? | ¿Clave? | Descripción                                                                                                                                                                                                                                                                                                                                                                                                                |
|-------------------|---------------|------------------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SECTNO            | INTEGER       | No                     | FK      | Número de sección en el paquete con el que está relacionada esta información de Explain.                                                                                                                                                                                                                                                                                                                                   |
| STREAM_ID         | INTEGER       | No                     | No      | ID exclusivo para esta corriente de datos en el operador especificado.                                                                                                                                                                                                                                                                                                                                                     |
| SOURCE_TYPE       | CHAR(1)       | No                     | No      | Indica la fuente de la corriente de datos:<br><b>O</b> Operador<br><b>D</b> Objeto de datos                                                                                                                                                                                                                                                                                                                                |
| SOURCE_ID         | SMALLINT      | No                     | No      | ID exclusivo para el operador dentro de esta consulta que es la fuente de esta corriente de datos. Se establece en -1 si SOURCE_TYPE es 'D'.                                                                                                                                                                                                                                                                               |
| TARGET_TYPE       | CHAR(1)       | No                     | No      | Indica el destino de la corriente de datos:<br><b>O</b> Operador<br><b>D</b> Objeto de datos                                                                                                                                                                                                                                                                                                                               |
| TARGET_ID         | SMALLINT      | No                     | No      | ID exclusivo para el operador dentro de esta consulta que es el destino de esta corriente de datos. Se establece en -1 si TARGET_TYPE es 'D'.                                                                                                                                                                                                                                                                              |
| OBJECT_SCHEMA     | VARCHAR(128)  | Sí                     | No      | El esquema al que pertenece el objeto de datos afectado. Se establece en nulo si SOURCE_TYPE y TARGET_TYPE son 'O'.                                                                                                                                                                                                                                                                                                        |
| OBJECT_NAME       | VARCHAR(128)  | Sí                     | No      | Nombre del objeto que es el sujeto de la corriente de datos. Se establece en nulo si SOURCE_TYPE y TARGET_TYPE son 'O'.                                                                                                                                                                                                                                                                                                    |
| STREAM_COUNT      | DOUBLE        | No                     | No      | Cardinalidad estimada de la corriente de datos.                                                                                                                                                                                                                                                                                                                                                                            |
| COLUMN_COUNT      | SMALLINT      | No                     | No      | Número de columnas en la corriente de datos.                                                                                                                                                                                                                                                                                                                                                                               |
| PREDICATE_ID      | INTEGER       | No                     | No      | Si la corriente forma parte de una subconsulta para un predicado, el ID del predicado se reflejará aquí, de lo contrario la columna se establece en -1.                                                                                                                                                                                                                                                                    |
| COLUMN_NAMES      | CLOB(1M)      | Sí                     | No      | Esta columna contiene los nombres y la información de ordenación de las columnas implicadas en esta corriente.<br><br>Estos nombres estarán en el formato de:<br>NOMBRE1 (A)+NOMBRE2 (D)+NOMBRE3+NOMBRE4<br><br>Donde (A) indica una columna por orden ascendente, (D) indica una columna por orden descendente y ninguna información de ordenación indica que la columna no está ordenada o que el orden no es relevante. |
| PMID              | SMALLINT      | No                     | No      | ID del mapa de particionamiento.                                                                                                                                                                                                                                                                                                                                                                                           |

Tabla 142. Tabla EXPLAIN\_STREAM (continuación)

| Nombre de columna | Tipo de datos | ¿Posibilidad de nulos? | ¿Clave? | Descripción                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-------------------|---------------|------------------------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SINGLE_NODE       | CHAR(5)       | Sí                     | No      | Indica si este flujo de datos está en una partición única o en varias particiones:<br><br><b>MULT</b> En varias particiones<br><b>COOR</b> En nodo del coordinador<br><b>HASH</b> Dirigido utilizando generación aleatoria<br><b>RID</b> Dirigido utilizando el ID de fila<br><b>FUNC</b> Dirigido utilizando una función (PARTITION() o NODENUMBER())<br><b>CORR</b> Dirigido utilizando un valor de correlación<br><br><b>Numeric</b><br>Dirigido hacia un nodo individual predeterminado |
| PARTITION_COLUMNS | CLOB(64K)     | Sí                     | No      | Lista de columnas en las que este flujo de datos está particionado.                                                                                                                                                                                                                                                                                                                                                                                                                         |

### Tabla ADVISE\_INDEX

La tabla ADVISE\_INDEX representa los índices recomendados.

Para ver la definición de esta tabla, consulte el apartado “Definición de tabla ADVISE\_INDEX” en la página 1491.

Tabla 143. Tabla ADVISE\_INDEX

| Nombre de columna | Tipo de datos | ¿Posibilidad de nulos? | ¿Clave? | Descripción                                                                                                                                                       |
|-------------------|---------------|------------------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EXPLAIN_REQUESTER | VARCHAR(128)  | No                     | No      | ID de autorización del iniciador de esta petición de Explain.                                                                                                     |
| EXPLAIN_TIME      | TIMESTAMP     | No                     | No      | Hora de inicio de la petición de Explain.                                                                                                                         |
| SOURCE_NAME       | VARCHAR(128)  | No                     | No      | Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estática. |
| SOURCE_SCHEMA     | VARCHAR(128)  | No                     | No      | Esquema, o calificador, de la fuente de la petición de Explain.                                                                                                   |
| EXPLAIN_LEVEL     | CHAR(1)       | No                     | No      | Nivel de información de Explain para el que esta fila es aplicable.                                                                                               |
| STMTNO            | INTEGER       | No                     | No      | Número de sentencia en el paquete con el que está relacionada esta información de Explain.                                                                        |

## Tablas de Explain

Tabla 143. Tabla *ADVISE\_INDEX* (continuación)

| Nombre de columna | Tipo de datos | ¿Posibilidad de nulos? | ¿Clave? | Descripción                                                                                                                                                                                                                                                                                                                                                                                |
|-------------------|---------------|------------------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SECTNO            | INTEGER       | No                     | No      | Número de sección en el paquete con el que está relacionada esta información de Explain.                                                                                                                                                                                                                                                                                                   |
| QUERYNO           | INTEGER       | No                     | No      | Identificador numérico para la sentencia de SQL explicada. Para sentencias de SQL dinámicas (excluyendo la sentencia EXPLAIN SQL) emitidas a través de CLP o CLI, el valor por omisión es un valor incrementado secuencialmente. De lo contrario, el valor por omisión es el valor de STMTNO para sentencias de SQL estáticas y 1 para sentencias de SQL dinámicas.                        |
| QUERYTAG          | CHAR(20)      | No                     | No      | Distintivo identificador para cada sentencia de SQL explicada. Para sentencias de SQL dinámicas emitidas a través de CLP (excluyendo la sentencia EXPLAIN SQL), el valor por omisión es 'CLP'. Para sentencias de SQL dinámicas emitidas a través de CLI (excluyendo la sentencia EXPLAIN SQL), el valor por omisión es 'CLI'. De lo contrario, el valor por omisión utilizado es blancos. |
| NAME              | VARCHAR(128)  | No                     | No      | Nombre del índice.                                                                                                                                                                                                                                                                                                                                                                         |
| CREATOR           | VARCHAR(128)  | No                     | No      | Calificador del nombre de índice.                                                                                                                                                                                                                                                                                                                                                          |
| TBNAME            | VARCHAR(128)  | No                     | No      | Nombre de la tabla o apodo en el que se define el índice.                                                                                                                                                                                                                                                                                                                                  |
| TBCREATOR         | VARCHAR(128)  | No                     | No      | Calificador del nombre de tabla.                                                                                                                                                                                                                                                                                                                                                           |
| COLNAMES          | CLOB(64K)     | No                     | No      | Lista de nombres de columnas.                                                                                                                                                                                                                                                                                                                                                              |
| UNIQUERULE        | CHAR(1)       | No                     | No      | Regla de unicidad:<br>D = Los duplicados están permitidos<br>P = Índice primario<br>U = Sólo se permiten entradas exclusivas                                                                                                                                                                                                                                                               |
| COLCOUNT          | SMALLINT      | No                     | No      | Número de columnas de la clave más el número de columnas INCLUDE si hay alguna.                                                                                                                                                                                                                                                                                                            |
| IID               | SMALLINT      | No                     | No      | ID interno de índice.                                                                                                                                                                                                                                                                                                                                                                      |
| NLEAF             | INTEGER       | No                     | No      | Número de páginas; -1 si no se reúnen estadísticas.                                                                                                                                                                                                                                                                                                                                        |
| NLEVELS           | SMALLINT      | No                     | No      | Número de niveles de índices; -1 si no se reúnen estadísticas.                                                                                                                                                                                                                                                                                                                             |
| FULLKEYCARD       | BIGINT        | No                     | No      | Número de valores de clave completa diferenciada; -1 si no se reúnen estadísticas.                                                                                                                                                                                                                                                                                                         |
| FIRSTKEYCARD      | BIGINT        | No                     | No      | Número de valores de primera clave diferenciada; -1 si no se reúnen estadísticas.                                                                                                                                                                                                                                                                                                          |

Tabla 143. Tabla ADVISE\_INDEX (continuación)

| Nombre de columna | Tipo de datos | ¿Posibilidad de nulos? | ¿Clave? | Descripción                                                                                                                                                                                                                                                                                                                                                                                               |
|-------------------|---------------|------------------------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CLUSTERRATIO      | SMALLINT      | No                     | No      | Grado de agrupación de los datos con el índice; -1 si no se reúnen estadísticas o si se reúnen estadísticas de índice detalladas (en cuyo caso, se utilizará CLUSTERFACTOR en su lugar).                                                                                                                                                                                                                  |
| CLUSTERFACTOR     | DOUBLE        | No                     | No      | Mejor medición del grado de agrupación o -1 si no se han reunido estadísticas de índice detalladas o si el índice está definido en un apodo.                                                                                                                                                                                                                                                              |
| USERDEFINED       | SMALLINT      | No                     | No      | Definido por el usuario.                                                                                                                                                                                                                                                                                                                                                                                  |
| SYSTEM_REQUIRED   | SMALLINT      | No                     | No      | 1 si este índice es necesario para la restricción de clave primaria o de clave de unicidad O BIEN si es el índice en la columna de identificador de objeto (OID) de una tabla con tipo.<br><br>2 si este índice es necesario para la restricción de clave primaria o de clave de unicidad Y es el índice en la columna de identificador de objeto (OID) de una tabla con tipo.<br><br>De lo contrario, 0. |
| CREATE_TIME       | TIMESTAMP     | No                     | No      | Hora en que se ha creado el índice.                                                                                                                                                                                                                                                                                                                                                                       |
| STATS_TIME        | TIMESTAMP     | Sí                     | No      | Última vez que se ha realizado un cambio en las estadísticas registradas para este índice. Nulo si no hay estadísticas disponibles.                                                                                                                                                                                                                                                                       |
| PAGE_FETCH_PAIRS  | VARCHAR(254)  | No                     | No      | Una lista de pares de enteros, representada en la forma de caracteres. Cada par representa el número de páginas de un almacenamiento intermedio hipotético y el número de lecturas de páginas necesario para explorar la tabla con este índice utilizando dicho almacenamiento intermedio hipotético. (Serie de longitud cero si no hay datos disponibles.)                                               |
| REMARKS           | VARCHAR(254)  | Sí                     | No      | Comentario suministrado por el usuario o nulo.                                                                                                                                                                                                                                                                                                                                                            |
| DEFINER           | VARCHAR(128)  | No                     | No      | Usuario que ha creado el índice.                                                                                                                                                                                                                                                                                                                                                                          |
| CONVERTED         | CHAR(1)       | No                     | No      | Reservado para su utilización en el futuro.                                                                                                                                                                                                                                                                                                                                                               |
| SEQUENTIAL_PAGES  | INTEGER       | No                     | No      | Número de páginas ubicadas en disco por orden de clave de índice con pocos o ningún vacío entre ellas. (-1 si no hay estadísticas disponibles.)                                                                                                                                                                                                                                                           |
| DENSITY           | INTEGER       | No                     | No      | Proporción de SEQUENTIAL_PAGES para numerar las páginas del rango de páginas ocupadas por el índice, expresada como un porcentaje (entero entre 0 y 100, -1 si no hay estadísticas disponibles.)                                                                                                                                                                                                          |
| FIRST2KEYCARD     | BIGINT        | No                     | No      | Número de claves diferenciadas que utilizan las dos primeras columnas del índice (-1 si no hay estadísticas o no son aplicables)                                                                                                                                                                                                                                                                          |

## Tablas de Explain

Tabla 143. Tabla ADVISE\_INDEX (continuación)

| Nombre de columna | Tipo de datos | ¿Posibilidad de nulos? | ¿Clave? | Descripción                                                                                                                                                                                       |
|-------------------|---------------|------------------------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FIRST3KEYCARD     | BIGINT        | No                     | No      | Número de claves diferenciadas que utilizan las tres primeras columnas del índice (-1 si no hay estadísticas o si no son aplicables)                                                              |
| FIRST4KEYCARD     | BIGINT        | No                     | No      | Número de claves diferenciadas que utilizan las cuatro primeras columnas del índice (-1 si no hay estadísticas o no son aplicables)                                                               |
| PCTFREE           | SMALLINT      | No                     | No      | Porcentaje de cada página de hoja de índice que se va a reservar durante la creación inicial del índice. Este espacio está disponible para inserciones futuras después de la creación del índice. |
| UNIQUE_COLCOUNT   | SMALLINT      | No                     | No      | Número de columnas necesarias para una clave de unicidad. Siempre <=COLCOUNT. < COLCOUNT solamente si hay columnas INCLUDE. -1 si el índice no tiene clave de unicidad (permite duplicados)       |
| MINPCTUSED        | SMALLINT      | No                     | No      | Si no es cero, entonces se habilita la reorganización de índices en línea y el valor es el umbral de espacio mínimo utilizado antes de fusionar las páginas.                                      |
| REVERSE_SCANS     | CHAR(1)       | No                     | No      | Y = El índice soporta exploraciones invertidas<br>N = El índice no soporta exploraciones invertidas                                                                                               |
| USE_INDEX         | CHAR(1)       | Sí                     | No      | Y = índice recomendado o evaluado<br>N = no se ha de recomendar índice                                                                                                                            |
| CREATION_TEXT     | CLOB(1M)      | No                     | No      | La sentencia de SQL se utiliza para crear el índice.                                                                                                                                              |
| PACKED_DESC       | BLOB(20M)     | Sí                     | No      | Descripción interna de la tabla.                                                                                                                                                                  |

### Tabla ADVISE\_WORKLOAD

La tabla ADVISE\_WORKLOAD representa la sentencia que forma la carga de trabajo. Para obtener mas detalles sobre la carga de trabajo consulte *Administration Guide: Performance*.

Para ver la definición de esta tabla, consulte el apartado "Definición de tabla ADVISE\_WORKLOAD" en la página 1493.



Tabla 144. Tabla ADVISE\_WORKLOAD

| Nombre de columna | Tipo de datos | ¿Posibilidad de nulos? | ¿Clave? | Descripción                                                                                              |
|-------------------|---------------|------------------------|---------|----------------------------------------------------------------------------------------------------------|
| WORKLOAD_NAME     | CHAR(128)     | No                     | No      | Nombre del conjunto de sentencias de SQL (carga de trabajo) a la que esta sentencia pertenece.           |
| STATEMENT_NO      | INTEGER       | No                     | No      | Número de sentencias de la carga de trabajo con el que está relacionada esta información de explicación. |
| STATEMENT_TEXT    | CLOB(1M)      | No                     | No      | Contenido de la sentencia de SQL.                                                                        |
| STATEMENT_TAG     | VARCHAR(256)  | No                     | No      | Distintivo identificador para cada sentencia de SQL explicada.                                           |
| FREQUENCY         | INTEGER       | No                     | No      | Las veces que esta sentencia aparece en la carga de trabajo.                                             |
| IMPORTANCE        | DOUBLE        | No                     | No      | Importancia de la sentencia.                                                                             |
| COST_BEFORE       | DOUBLE        | Sí                     | No      | El coste (en timerons) de la consulta si no se crean los índices recomendados.                           |
| COST_AFTER        | DOUBLE        | Sí                     | No      | El coste (en timerons) de la consulta si se crean los índices recomendados.                              |

### Definiciones de tabla para tablas de Explain

Las tablas de Explain se deben crear antes de invocar Explain. Las definiciones siguientes especifican cómo crear las Tablas de Explain necesarias:

- “Definición de tabla EXPLAIN\_ARGUMENT” en la página 1484
- “Definición de tabla EXPLAIN\_INSTANCE” en la página 1485
- “Definición de tabla EXPLAIN\_OBJECT” en la página 1486
- “Definición de tabla EXPLAIN\_OPERATOR” en la página 1487
- “Definición de tabla EXPLAIN\_PREDICATE” en la página 1488
- “Definición de tabla EXPLAIN\_STATEMENT” en la página 1489
- “Definición de tabla EXPLAIN\_STREAM” en la página 1490
- “Definición de tabla ADVISE\_INDEX” en la página 1491
- “Definición de tabla ADVISE\_WORKLOAD” en la página 1493

De manera alternativa, créelas utilizando el script de entrada del procesador de línea de mandatos de muestra, proporcionado en el archivo EXPLAIN.DDL situado en el subdirectorio 'misc' del directorio 'sqlib'. Conéctese a la base de datos donde se necesitan las tablas de Explain. Emita el mandato: db2 -tf EXPLAIN.DDL y se crearán las tablas.

## Tablas de Explain

### Definición de tabla EXPLAIN\_ARGUMENT

```
CREATE TABLE EXPLAIN_ARGUMENT (EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
 EXPLAIN_TIME TIMESTAMP NOT NULL,
 SOURCE_NAME VARCHAR(128) NOT NULL,
 SOURCE_SCHEMA VARCHAR(128) NOT NULL,
 EXPLAIN_LEVEL CHAR(1) NOT NULL,
 STMTNO INTEGER NOT NULL,
 SECTNO INTEGER NOT NULL,
 OPERATOR_ID INTEGER NOT NULL,
 ARGUMENT_TYPE CHAR(8) NOT NULL,
 ARGUMENT_VALUE VARCHAR(1024) NOT NULL,
 LONG_ARGUMENT_VALUE CLOB(1M) NOT LOGGED,
 FOREIGN KEY (EXPLAIN_REQUESTER,
 EXPLAIN_TIME,
 SOURCE_NAME,
 SOURCE_SCHEMA,
 EXPLAIN_LEVEL,
 STMTNO,
 SECTNO)
 REFERENCES EXPLAIN_STATEMENT
 ON DELETE CASCADE)
```

## Definición de tabla EXPLAIN\_INSTANCE

```

CREATE TABLE EXPLAIN_INSTANCE (EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
 EXPLAIN_TIME TIMESTAMP NOT NULL,
 SOURCE_NAME VARCHAR(128) NOT NULL,
 SOURCE_SCHEMA VARCHAR(128) NOT NULL,
 EXPLAIN_OPTION CHAR(1) NOT NULL,
 SNAPSHOT_TAKEN CHAR(1) NOT NULL,
 DB2_VERSION CHAR(7) NOT NULL,
 SQL_TYPE CHAR(1) NOT NULL,
 QUERYOPT INTEGER NOT NULL,
 BLOCK CHAR(1) NOT NULL,
 ISOLATION CHAR(2) NOT NULL,
 BUFFPAGE INTEGER NOT NULL,
 AVG_APPLS INTEGER NOT NULL,
 SORTHEAP INTEGER NOT NULL,
 LOCKLIST INTEGER NOT NULL,
 MAXLOCKS SMALLINT NOT NULL,
 LOCKS_AVAIL INTEGER NOT NULL,
 CPU_SPEED DOUBLE NOT NULL,
 REMARKS VARCHAR(254),
 DBHEAP INTEGER NOT NULL,
 COMM_SPEED DOUBLE NOT NULL,
 PARALLELISM CHAR(2) NOT NULL,
 DATAJOINER CHAR(1) NOT NULL,
 PRIMARY KEY (EXPLAIN_REQUESTER,
 EXPLAIN_TIME,
 SOURCE_NAME,
 SOURCE_SCHEMA))

```

## Tablas de Explain

### Definición de tabla EXPLAIN\_OBJECT

```
CREATE TABLE EXPLAIN_OBJECT (EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
 EXPLAIN_TIME TIMESTAMP NOT NULL,
 SOURCE_NAME VARCHAR(128) NOT NULL,
 SOURCE_SCHEMA VARCHAR(128) NOT NULL,
 EXPLAIN_LEVEL CHAR(1) NOT NULL,
 STMTNO INTEGER NOT NULL,
 SECTNO INTEGER NOT NULL,
 OBJECT_SCHEMA VARCHAR(128) NOT NULL,
 OBJECT_NAME VARCHAR(128) NOT NULL,
 OBJECT_TYPE CHAR(2) NOT NULL,
 CREATE_TIME TIMESTAMP,
 STATISTICS_TIME TIMESTAMP,
 COLUMN_COUNT SMALLINT NOT NULL,
 ROW_COUNT INTEGER NOT NULL,
 WIDTH INTEGER NOT NULL,
 PAGES INTEGER NOT NULL,
 DISTINCT CHAR(1) NOT NULL,
 TABLESPACE_NAME VARCHAR(128),
 OVERHEAD DOUBLE NOT NULL,
 TRANSFER_RATE DOUBLE NOT NULL,
 PREFETCHSIZE INTEGER NOT NULL,
 EXTENTSIZE INTEGER NOT NULL,
 CLUSTER DOUBLE NOT NULL,
 NLEAF INTEGER NOT NULL,
 NLEVELS INTEGER NOT NULL,
 FULLKEYCARD BIGINT NOT NULL,
 OVERFLOW INTEGER NOT NULL,
 FIRSTKEYCARD BIGINT NOT NULL,
 FIRST2KEYCARD BIGINT NOT NULL,
 FIRST3KEYCARD BIGINT NOT NULL,
 FIRST4KEYCARD BIGINT NOT NULL,
 SEQUENTIAL_PAGES INTEGER NOT NULL,
 DENSITY INTEGER NOT NULL,
 FOREIGN KEY (EXPLAIN_REQUESTER,
 EXPLAIN_TIME,
 SOURCE_NAME,
 SOURCE_SCHEMA,
 EXPLAIN_LEVEL,
 STMTNO,
 SECTNO)
 REFERENCES EXPLAIN_STATEMENT
 ON DELETE CASCADE)
```

## Definición de tabla EXPLAIN\_OPERATOR

```

CREATE TABLE EXPLAIN_OPERATOR (EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
 EXPLAIN_TIME TIMESTAMP NOT NULL,
 SOURCE_NAME VARCHAR(128) NOT NULL,
 SOURCE_SCHEMA VARCHAR(128) NOT NULL,
 EXPLAIN_LEVEL CHAR(1) NOT NULL,
 STMTNO INTEGER NOT NULL,
 SECTNO INTEGER NOT NULL,
 OPERATOR_ID INTEGER NOT NULL,
 OPERATOR_TYPE CHAR(6) NOT NULL,
 TOTAL_COST DOUBLE NOT NULL,
 IO_COST DOUBLE NOT NULL,
 CPU_COST DOUBLE NOT NULL,
 FIRST_ROW_COST DOUBLE NOT NULL,
 RE_TOTAL_COST DOUBLE NOT NULL,
 RE_IO_COST DOUBLE NOT NULL,
 RE_CPU_COST DOUBLE NOT NULL,
 COMM_COST DOUBLE NOT NULL,
 FIRST_COMM_COST DOUBLE NOT NULL,
 REMOTE_TOTAL_COST DOUBLE NOT NULL,
 REMOTE_COMM_COST DOUBLE NOT NULL,
 FOREIGN KEY (EXPLAIN_REQUESTER,
 EXPLAIN_TIME,
 SOURCE_NAME,
 SOURCE_SCHEMA,
 EXPLAIN_LEVEL,
 STMTNO,
 SECTNO)
 REFERENCES EXPLAIN_STATEMENT
 ON DELETE CASCADE)

```

## Tablas de Explain

### Definición de tabla EXPLAIN\_PREDICATE

```
CREATE TABLE EXPLAIN_PREDICATE (EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
EXPLAIN_TIME TIMESTAMP NOT NULL,
SOURCE_NAME VARCHAR(128) NOT NULL,
SOURCE_SCHEMA VARCHAR(128) NOT NULL,
EXPLAIN_LEVEL CHAR(1) NOT NULL,
STMTNO INTEGER NOT NULL,
SECTNO INTEGER NOT NULL,
OPERATOR_ID INTEGER NOT NULL,
PREDICATE_ID INTEGER NOT NULL,
HOW_APPLIED CHAR(5) NOT NULL,
WHEN_EVALUATED CHAR(3) NOT NULL,
RELOP_TYPE CHAR(2) NOT NULL,
SUBQUERY CHAR(1) NOT NULL,
FILTER_FACTOR DOUBLE NOT NULL,
PREDICATE_TEXT CLOB(1M) NOT LOGGED,
FOREIGN KEY (EXPLAIN_REQUESTER,
EXPLAIN_TIME,
SOURCE_NAME,
SOURCE_SCHEMA,
EXPLAIN_LEVEL,
STMTNO,
SECTNO)
REFERENCES EXPLAIN_STATEMENT
ON DELETE CASCADE)
```

**Definición de tabla EXPLAIN\_STATEMENT**

```

CREATE TABLE EXPLAIN_STATEMENT (EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
 EXPLAIN_TIME TIMESTAMP NOT NULL,
 SOURCE_NAME VARCHAR(128) NOT NULL,
 SOURCE_SCHEMA VARCHAR(128) NOT NULL,
 EXPLAIN_LEVEL CHAR(1) NOT NULL,
 STMTNO INTEGER NOT NULL,
 SECTNO INTEGER NOT NULL,
 QUERYNO INTEGER NOT NULL,
 QUERYTAG CHAR(20) NOT NULL,
 STATEMENT_TYPE CHAR(2) NOT NULL,
 UPDATABLE CHAR(1) NOT NULL,
 DELETABLE CHAR(1) NOT NULL
TOTAL_COST DOUBLE NOT NULL,
 STATEMENT_TEXT CLOB(1M) NOT NULL
 NOT LOGGED,
 SNAPSHOT BLOB(10M) NOT LOGGED,
 QUERY_DEGREE INTEGER NOT NULL,
 PRIMARY KEY (EXPLAIN_REQUESTER,
 EXPLAIN_TIME,
 SOURCE_NAME,
 SOURCE_SCHEMA,
 EXPLAIN_LEVEL,
 STMTNO,
 SECTNO),
 FOREIGN KEY (EXPLAIN_REQUESTER,
 EXPLAIN_TIME,
 SOURCE_NAME,
 SOURCE_SCHEMA)
 REFERENCES EXPLAIN_INSTANCE
ON DELETE CASCADE)

```

## Tablas de Explain

### Definición de tabla EXPLAIN\_STREAM

```
CREATE TABLE EXPLAIN_STREAM (EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
 EXPLAIN_TIME TIMESTAMP NOT NULL,
 SOURCE_NAME VARCHAR(128) NOT NULL,
 SOURCE_SCHEMA VARCHAR(128) NOT NULL,
 EXPLAIN_LEVEL CHAR(1) NOT NULL,
 STMTNO INTEGER NOT NULL,
 SECTNO INTEGER NOT NULL,
 STREAM_ID INTEGER NOT NULL,
 SOURCE_TYPE CHAR(1) NOT NULL,
 SOURCE_ID SMALLINT NOT NULL,
 TARGET_TYPE CHAR(1) NOT NULL,
 TARGET_ID SMALLINT NOT NULL,
 OBJECT_SCHEMA VARCHAR(128),
 OBJECT_NAME VARCHAR(128),
 STREAM_COUNT DOUBLE NOT NULL,
 COLUMN_COUNT SMALLINT NOT NULL,
 PREDICATE_ID INTEGER NOT NULL,
 COLUMN_NAMES CLOB(1M) NOT LOGGED,
 PMID SMALLINT NOT NULL,
 SINGLE_NODE CHAR(5),
 PARTITION_COLUMNS CLOB(64K) NOT LOGGED,
 FOREIGN KEY (EXPLAIN_REQUESTER,
 EXPLAIN_TIME,
 SOURCE_NAME,
 SOURCE_SCHEMA,
 EXPLAIN_LEVEL,
 STMTNO,
 SECTNO)
 REFERENCES EXPLAIN_STATEMENT
 ON DELETE CASCADE)
```



**Definición de tabla ADVISE\_INDEX**

```

CREATE TABLE ADVISE_INDEX (EXPLAIN_REQUESTER VARCHAR(128) NOT NULL
 WITH DEFAULT '',
 EXPLAIN_TIME TIMESTAMP NOT NULL
 WITH DEFAULT CURRENT_TIMESTAMP,
 SOURCE_NAME VARCHAR(128) NOT NULL
 WITH DEFAULT '',
 SOURCE_SCHEMA VARCHAR(128) NOT NULL
 WITH DEFAULT '',
 EXPLAIN_LEVEL CHAR(1) NOT NULL
 WITH DEFAULT '',
 STMTNO INTEGER NOT NULL
 WITH DEFAULT 0,
 SECTNO INTEGER NOT NULL
 WITH DEFAULT 0,
 QUERYNO INTEGER NOT NULL
 WITH DEFAULT 0,
 QUERYTAG CHAR(20) NOT NULL
 WITH DEFAULT '',
 NAME VARCHAR(128) NOT NULL,
 CREATOR VARCHAR(128) NOT NULL
 WITH DEFAULT '',
 TBNAME VARCHAR(128) NOT NULL,
 TBCREATOR VARCHAR(128) NOT NULL
 WITH DEFAULT '',
 COLNAMES CLOB(64K) NOT NULL,
 UNIQUERULE CHAR(1) NOT NULL
 WITH DEFAULT '',
 COLCOUNT SMALLINT NOT NULL
 WITH DEFAULT 0,
 IID SMALLINT NOT NULL
 WITH DEFAULT 0,
 NLEAF INTEGER NOT NULL
 WITH DEFAULT 0,
 NLEVELS SMALLINT NOT NULL
 WITH DEFAULT 0,
 FIRSTKEYCARD BIGINT NOT NULL
 WITH DEFAULT 0,
 FULLKEYCARD BIGINT NOT NULL
 WITH DEFAULT 0,
 CLUSTERRATIO SMALLINT NOT NULL
 WITH DEFAULT 0,
 CLUSTERFACTOR DOUBLE NOT NULL
 WITH DEFAULT 0,
 USERDEFINED SMALLINT NOT NULL
 WITH DEFAULT 0,
 SYSTEM_REQUIRED SMALLINT NOT NULL
 WITH DEFAULT 0,
 CREATE_TIME TIMESTAMP NOT NULL
 WITH DEFAULT CURRENT_TIMESTAMP,
 STATS_TIME TIMESTAMP
 WITH DEFAULT CURRENT_TIMESTAMP,
 PAGE_FETCH_PAIRS VARCHAR(254) NOT NULL
 WITH DEFAULT '',
 REMARKS VARCHAR(254)

```

## Tablas de Explain

|                  |                                                              |
|------------------|--------------------------------------------------------------|
| DEFINER          | WITH DEFAULT '',<br>VARCHAR(128) NOT NULL<br>WITH DEFAULT '' |
| CONVERTED        | CHAR(1) NOT NULL<br>WITH DEFAULT ''                          |
| SEQUENTIAL_PAGES | INTEGER NOT NULL<br>WITH DEFAULT 0                           |
| DENSITY          | INTEGER NOT NULL<br>WITH DEFAULT 0                           |
| FIRST2KEYCARD    | BIGINT NOT NULL<br>WITH DEFAULT 0                            |
| FIRST3KEYCARD    | BIGINT NOT NULL<br>WITH DEFAULT 0                            |
| FIRST4KEYCARD    | BIGINT NOT NULL<br>WITH DEFAULT 0                            |
| PCTFREE          | SMALLINT NOT NULL<br>WITH DEFAULT -1                         |
| UNIQUE_COLCOUNT  | SMALLINT NOT NULL<br>WITH DEFAULT -1                         |
| MINPCTUSED       | SMALLINT NOT NULL<br>WITH DEFAULT 0                          |
| REVERSE_SCANS    | CHAR(1) NOT NULL<br>WITH DEFAULT 'N'                         |
| USE_INDEX        | CHAR(1)                                                      |
| CREATION_TEXT    | CLOB(1M) NOT NULL<br>NOT LOGGED WITH DEFAULT ''              |
| PACKED_DESC      | BLOB(1M) NOT LOGGED)                                         |

**Definición de tabla ADVISE\_WORKLOAD**

```
CREATE TABLE ADVISE_WORKLOAD (WORKLOAD_NAME CHAR(128) NOT NULL
 WITH DEFAULT 'WK0',
 STATEMENT_NO INTEGER NOT NULL
 WITH DEFAULT 1,
 STATEMENT_TEXT CLOB(1M) NOT NULL NOT LOGGED,
 STATEMENT_TAG VARCHAR(256) NOT NULL
 WITH DEFAULT '',
 FREQUENCY INTEGER NOT NULL
 WITH DEFAULT 1,
 IMPORTANCE DOUBLE NOT NULL
 WITH DEFAULT 1,
 COST_BEFORE DOUBLE,
 COST_AFTER DOUBLE)
```

## Tablas de Explain

## Apéndice L. Valores de registro de explicaciones

Este apéndice describe la interacción de los valores especiales del registro CURRENT EXPLAIN MODE y CURRENT EXPLAIN SNAPSHOT entre ellos y con los mandatos PREP y BIND.

Los valores de los registros especiales CURRENT EXPLAIN MODE y CURRENT EXPLAIN SNAPSHOT interactúan de la siguiente manera en el SQL dinámico.

Tabla 145. Interacción de los valores de registro especial de explicación para SQL dinámico

| Valores de EXPLAIN SNAPSHOT | Valores de EXPLAIN MODE                                                                                                                        |                                                                                                                                                                                            |                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                          |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                             | NO                                                                                                                                             | YES                                                                                                                                                                                        | EXPLAIN                                                                                                                                                                                                                                 | RECOMMEND INDEXES                                                                                                                                                                                                                                                        | EVALUATE INDEXES                                                                                                                                                                                                                                                         |
| NO                          | <ul style="list-style-type: none"> <li>Se devuelven los resultados de la consulta.</li> </ul>                                                  | <ul style="list-style-type: none"> <li>Se llenan las tablas de explicación</li> <li>Se devuelven los resultados de la consulta.</li> </ul>                                                 | <ul style="list-style-type: none"> <li>Se llenan las tablas de explicación.</li> <li>Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).</li> </ul>                                                | <ul style="list-style-type: none"> <li>Se llenan las tablas de explicación.</li> <li>Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).</li> <li>Se recomiendan índices.</li> </ul>                                                | <ul style="list-style-type: none"> <li>Se llenan las tablas de explicación.</li> <li>Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).</li> <li>Se evalúan los índices.</li> </ul>                                                |
| YES                         | <ul style="list-style-type: none"> <li>Se toma una Instantánea de explicación.</li> <li>Se devuelven los resultados de la consulta.</li> </ul> | <ul style="list-style-type: none"> <li>Se llenan las tablas de explicación</li> <li>Se toma una instantánea de explicación</li> <li>Se devuelven los resultados de la consulta.</li> </ul> | <ul style="list-style-type: none"> <li>Se llenan las tablas de explicación</li> <li>Se toma una instantánea de explicación</li> <li>Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).</li> </ul> | <ul style="list-style-type: none"> <li>Se llenan las tablas de explicación</li> <li>Se toma una instantánea de explicación</li> <li>Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).</li> <li>Se recomiendan índices.</li> </ul> | <ul style="list-style-type: none"> <li>Se llenan las tablas de explicación</li> <li>Se toma una instantánea de explicación</li> <li>Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).</li> <li>Se evalúan los índices.</li> </ul> |

Tabla 145. Interacción de los valores de registro especial de explicación para SQL dinámico (continuación)

| Valores de EXPLAIN SNAPSHOT | Valores de EXPLAIN MODE                                                                                                                                            |                                                                                                                                                                                                                 |                                                                                                                                                                                                                 |                                                                                                                                                                                                                                                  |                                                                                                                                                                                                                                                  |
|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                             | NO                                                                                                                                                                 | YES                                                                                                                                                                                                             | EXPLAIN                                                                                                                                                                                                         | RECOMMEND INDEXES                                                                                                                                                                                                                                | EVALUATE INDEXES                                                                                                                                                                                                                                 |
| EXPLAIN                     | <ul style="list-style-type: none"> <li>Se toma una instantánea de explicación de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).</li> </ul> | <ul style="list-style-type: none"> <li>Se llenan las tablas de explicación</li> <li>Se toma una instantánea de explicación de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).</li> </ul> | <ul style="list-style-type: none"> <li>Se llenan las tablas de explicación</li> <li>Se toma una instantánea de explicación de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).</li> </ul> | <ul style="list-style-type: none"> <li>Se llenan las tablas de explicación</li> <li>Se toma una instantánea de explicación de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).</li> <li>Se recomiendan índices.</li> </ul> | <ul style="list-style-type: none"> <li>Se llenan las tablas de explicación</li> <li>Se toma una instantánea de explicación de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).</li> <li>Se evalúan los índices.</li> </ul> |

El registro especial CURRENT EXPLAIN MODE interactúa con la opción de enlace lógico EXPLAIN de la siguiente manera en el SQL dinámico.

Tabla 146. Interacción de la opción de enlace lógico EXPLAIN y CURRENT EXPLAIN MODE

| Valores de EXPLAIN MODE | Valores de la opción de enlace lógico EXPLAIN                                                                                                                   |                                                                                                                                                                                                                                  |                                                                                                                                                                                                                                  |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                         | NO                                                                                                                                                              | YES                                                                                                                                                                                                                              | ALL                                                                                                                                                                                                                              |
| NO                      | <ul style="list-style-type: none"> <li>Se devuelven los resultados de la consulta.</li> </ul>                                                                   | <ul style="list-style-type: none"> <li>Las tablas de explicación sellenan para el SQL estático</li> <li>Se devuelven los resultados de la consulta.</li> </ul>                                                                   | <ul style="list-style-type: none"> <li>Las tablas de explicación sellenan para el SQL estático</li> <li>Las tablas de explicación se llenan para el SQL dinámico</li> <li>Se devuelven los resultados de la consulta.</li> </ul> |
| YES                     | <ul style="list-style-type: none"> <li>Las tablas de explicación se llenan para el SQL dinámico</li> <li>Se devuelven los resultados de la consulta.</li> </ul> | <ul style="list-style-type: none"> <li>Las tablas de explicación sellenan para el SQL estático</li> <li>Las tablas de explicación se llenan para el SQL dinámico</li> <li>Se devuelven los resultados de la consulta.</li> </ul> | <ul style="list-style-type: none"> <li>Las tablas de explicación sellenan para el SQL estático</li> <li>Las tablas de explicación se llenan para el SQL dinámico</li> <li>Se devuelven los resultados de la consulta.</li> </ul> |

Tabla 146. Interacción de la opción de enlace lógico EXPLAIN y CURRENT EXPLAIN MODE (continuación)

| Valores de EXPLAIN MODE | Valores de la opción de enlace lógico EXPLAIN                                                                                                                                                                                                |                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                               |
|-------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                         | NO                                                                                                                                                                                                                                           | YES                                                                                                                                                                                                                                                                                                           | ALL                                                                                                                                                                                                                                                                                                           |
| EXPLAIN                 | <ul style="list-style-type: none"> <li>Las tablas de explicación se llenan para el SQL dinámico</li> <li>Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).</li> </ul>                                 | <ul style="list-style-type: none"> <li>Las tablas de explicación sellenan para el SQL estático</li> <li>Las tablas de explicación se llenan para el SQL dinámico</li> <li>Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).</li> </ul>                                 | <ul style="list-style-type: none"> <li>Las tablas de explicación sellenan para el SQL estático</li> <li>Las tablas de explicación se llenan para el SQL dinámico</li> <li>Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).</li> </ul>                                 |
| RECOMMEND INDEXES       | <ul style="list-style-type: none"> <li>Las tablas de explicación se llenan para el SQL dinámico</li> <li>Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).</li> <li>Se recomiendan índices</li> </ul> | <ul style="list-style-type: none"> <li>Las tablas de explicación sellenan para el SQL estático</li> <li>Las tablas de explicación se llenan para el SQL dinámico</li> <li>Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).</li> <li>Se recomiendan índices</li> </ul> | <ul style="list-style-type: none"> <li>Las tablas de explicación sellenan para el SQL estático</li> <li>Las tablas de explicación se llenan para el SQL dinámico</li> <li>Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).</li> <li>Se recomiendan índices</li> </ul> |
| EVALUATE INDEXES        | <ul style="list-style-type: none"> <li>Las tablas de explicación se llenan para el SQL dinámico</li> <li>Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).</li> <li>Se evalúan los índices</li> </ul> | <ul style="list-style-type: none"> <li>Las tablas de explicación sellenan para el SQL estático</li> <li>Las tablas de explicación se llenan para el SQL dinámico</li> <li>Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).</li> <li>Se evalúan los índices</li> </ul> | <ul style="list-style-type: none"> <li>Las tablas de explicación sellenan para el SQL estático</li> <li>Las tablas de explicación se llenan para el SQL dinámico</li> <li>Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).</li> <li>Se evalúan los índices</li> </ul> |

El registro especial CURRENT EXPLAIN SNAPSHOT interactúa con la opción de enlace lógico EXPLSNAP de la siguiente manera para el SQL dinámico.

Tabla 147. Interacción de la opción de enlace EXPLSNAP y CURRENT EXPLAIN SNAPSHOT

| Valores de EXPLAIN SNAPSHOT | Valores de la opción de enlace EXPLSNAP                                                                                                                                                                         |                                                                                                                                                                                                                                                                                      |                                                                                                                                                                                                                                                                                      |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                             | NO                                                                                                                                                                                                              | YES                                                                                                                                                                                                                                                                                  | ALL                                                                                                                                                                                                                                                                                  |
| NO                          | <ul style="list-style-type: none"> <li>Se devuelven los resultados de la consulta.</li> </ul>                                                                                                                   | <ul style="list-style-type: none"> <li>Se toma una Instantánea de explicación para el SQL estático</li> <li>Se devuelven los resultados de la consulta.</li> </ul>                                                                                                                   | <ul style="list-style-type: none"> <li>Se toma una Instantánea de explicación para el SQL estático</li> <li>Se toma una Instantánea de explicación para el SQL dinámico</li> <li>Se devuelven los resultados de la consulta.</li> </ul>                                              |
| YES                         | <ul style="list-style-type: none"> <li>Se toma una Instantánea de explicación para el SQL dinámico</li> <li>Se devuelven los resultados de la consulta.</li> </ul>                                              | <ul style="list-style-type: none"> <li>Se toma una Instantánea de explicación para el SQL estático</li> <li>Se toma una Instantánea de explicación para el SQL dinámico</li> <li>Se devuelven los resultados de la consulta.</li> </ul>                                              | <ul style="list-style-type: none"> <li>Se toma una Instantánea de explicación para el SQL estático</li> <li>Se toma una Instantánea de explicación para el SQL dinámico</li> <li>Se devuelven los resultados de la consulta.</li> </ul>                                              |
| EXPLAIN                     | <ul style="list-style-type: none"> <li>Se toma una Instantánea de explicación para el SQL dinámico</li> <li>Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).</li> </ul> | <ul style="list-style-type: none"> <li>Se toma una Instantánea de explicación para el SQL estático</li> <li>Se toma una Instantánea de explicación para el SQL dinámico</li> <li>Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).</li> </ul> | <ul style="list-style-type: none"> <li>Se toma una Instantánea de explicación para el SQL estático</li> <li>Se toma una Instantánea de explicación para el SQL dinámico</li> <li>Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).</li> </ul> |



---

## Apéndice M. Ejemplo de recurrencia: Lista de material

Las aplicaciones de tipo Lista de material (BOM) son una necesidad habitual en muchos entornos comerciales. Para ilustrar la capacidad de una expresión de tabla común recursiva para aplicaciones BOM, considere una tabla de piezas con subpiezas asociadas y la cantidad de subpiezas que se precisan en la pieza. Para este ejemplo, cree la tabla como se muestra a continuación.

```
CREATE TABLE PARTLIST
(PIEZA VARCHAR(8),
SUBPART VARCHAR(8),
CANTIDAD INTEGER);
```

Para obtener resultados de consulta en este ejemplo, supongamos que la tabla LISTA DE PIEZAS se compone de los siguientes valores.

| PIEZA | SUBPIEZA | CANTIDAD |
|-------|----------|----------|
| 00    | 01       | 5        |
| 00    | 05       | 3        |
| 01    | 02       | 2        |
| 01    | 03       | 3        |
| 01    | 04       | 4        |
| 01    | 06       | 3        |
| 02    | 05       | 7        |
| 02    | 06       | 6        |
| 03    | 07       | 6        |
| 04    | 08       | 10       |
| 04    | 09       | 11       |
| 05    | 10       | 10       |
| 05    | 11       | 10       |
| 06    | 12       | 10       |
| 06    | 13       | 10       |
| 07    | 14       | 8        |
| 07    | 12       | 8        |

---

### Ejemplo 1: Explosión de primer nivel

El primer ejemplo se denomina explosión de primer nivel. Responde a la pregunta "¿Qué piezas son necesarias para crear la pieza identificada mediante '01'?". La lista incluirá las subpiezas directas, subpiezas de subpiezas, etc. Sin embargo, si una pieza se utiliza varias veces, las subpiezas correspondientes sólo aparecerán en la lista una vez.

```
WITH RPL (PIEZA, SUBPIEZA, CANTIDAD) AS
(SELECT PIEZA.RAIZ, SUBPIEZA.RAIZ, CANTIDAD.RAIZ
 FROM LISTA DE PIEZAS RAIZ
 WHERE PIEZA.RAIZ = '01'
 UNION ALL
 SELECT PIEZA.HIJA, SUBPIEZA.HIJA, CANTIDAD.HIJA
```

## Ejemplo de recurrencia: Lista de material

```
 FROM RPL PADRE, LISTA DE PIEZAS HIJA
 WHERE SUBPIEZA.PADRE = PIEZA.HIJA
)
SELECT DISTINCT PIEZA, SUBPIEZA, CANTIDAD
FROM RPL
ORDER BY PIEZA, SUBPIEZA, CANTIDAD;
```

La consulta anterior incluye una expresión de tabla común, identificada mediante el nombre *RPL*, que expresa la pieza repetitiva de esta consulta. Ilustra los elementos básicos de una expresión de tabla común recursiva.

El primer operando (selección completa) de la UNION, al que se hace referencia como la *selección completa de inicialización*, obtiene los hijos directos de la pieza '01'. La cláusula FROM de esta selección completa hace referencia a la tabla fuente y nunca se hará referencia a sí misma (*RPL* en este caso). El resultado de la primera selección completa va a la expresión de tabla común *RPL* (LISTA DE PIEZAS recursiva). Como en este ejemplo, UNION debe ser siempre UNION ALL.

El segundo operando (selección completa) de UNION utiliza *RPL* para calcular las subpiezas de subpiezas haciendo que la cláusula FROM haga referencia a la expresión de tabla común *RPL* y la tabla fuente con una unión de una pieza de la tabla fuente (hija) a una subpieza del resultado actual contenido en *RPL* (padre). El resultado vuelve a *RPL* de nuevo. El segundo operando de UNION se utiliza entonces repetidamente hasta que ya no existan más hijas.

SELECT DISTINCT de la selección completa principal de esta consulta, garantiza que no aparezca en la lista la misma pieza/subpieza más de una vez.

El resultado de la consulta es como sigue:

| PIEZA | SUBPIEZA | CANTIDAD |
|-------|----------|----------|
| 01    | 02       | 2        |
| 01    | 03       | 3        |
| 01    | 04       | 4        |
| 01    | 06       | 3        |
| 02    | 05       | 7        |
| 02    | 06       | 6        |
| 03    | 07       | 6        |
| 04    | 08       | 10       |
| 04    | 09       | 11       |
| 05    | 10       | 10       |
| 05    | 11       | 10       |
| 06    | 12       | 10       |
| 06    | 13       | 10       |
| 07    | 12       | 8        |
| 07    | 14       | 8        |

## Ejemplo de recurrencia: Lista de material

Observe, en el resultado, que de la pieza '01' se pasa a la pieza '02', que a su vez pasa a la '06', etc. Observe también que la pieza '06' se alcanza dos veces, una a través de '01' directamente y otra a través de '02'. En el resultado, sin embargo, los subcomponentes sólo aparecen una vez en la lista (es el resultado de utilizar SELECT DISTINCT), tal como se requiere.

Es importante recordar que con las expresiones de tabla comunes recursivas puede generarse un *bucle infinito*. En este ejemplo, se produciría un bucle infinito si la condición de búsqueda del segundo operando que une las tablas madre e hija tuviera esta codificación:

```
SUBPIEZA.PADRE = SUBPIEZA.HIJA
```

Este ejemplo de bucle infinito es consecuencia de no codificar lo que se intenta codificar. Sin embargo, debe extremar la precaución al determinar qué es lo que se ha de codificar, de forma que se consiga un final definitivo del ciclo de recurrencia.

El resultado de esta consulta de ejemplo puede producirse en un programa de aplicación sin utilizar una expresión de tabla común recursiva. Sin embargo, ello requeriría iniciar una nueva consulta para cada nivel de repetición. Además, la aplicación necesita colocar de nuevo todos los resultados en la base de datos para ordenar el resultado. Todo ello hace que la lógica de la aplicación se complique y que el funcionamiento no sea el esperado. La lógica de la aplicación resulta aún más complicada e ineficiente para consultas de otras listas de material, tales como consultas resumidas y de explosión.

---

### Ejemplo 2: Explosión resumida

El segundo ejemplo es una explosión resumida. La cuestión que se plantea aquí es la cantidad total de cada pieza que se requiere para crear la pieza '01'. La diferencia principal de la explosión de un solo nivel es la necesidad de agregar las cantidades. El primer ejemplo indica la cantidad de subpiezas necesarias para la pieza siempre que se requiera. No indica cuántas de las subpiezas se necesitan para crear la pieza '01'.

```
WITH RPL (PIEZA, SUBPIEZA, CANTIDAD) AS
(
 SELECT PIEZA.RAIZ, SUBPIEZA.RAIZ, CANTIDAD.RAIZ
 FROM LISTA DE PIEZAS RAIZ
 WHERE PIEZA.RAIZ = '01'
 UNION ALL
 SELECT PIEZA.PADRE, SUBPIEZA.HIJA, CANTIDAD.PADRE*CANTIDAD.HIJA
 FROM RPL PADRE, LISTA DE PIEZAS HIJA
 WHERE SUBPIEZA.PADRE = PIEZA.HIJA
)
SELECT PIEZA, SUBPIEZA, SUM(CANTIDAD) AS "CANT. total usada"
FROM RPL
GROUP BY PIEZA, SUBPIEZA
ORDER BY PIEZA, SUBPIEZA;
```

## Ejemplo de recurrencia: Lista de material

En la consulta anterior, la lista de selección del segundo operando de UNION en la expresión de tabla común recursiva, identificada mediante el nombre *RPL*, muestra la agregación de la cantidad. Para averiguar qué porcentaje de subpieza se utiliza, la cantidad del elemento madre se multiplica por la cantidad por madre de una hija. Si una pieza se utiliza varias veces en lugares diferentes, requerirá otra agregación final. Esto se realiza por la agrupación por la expresión de tabla común *RPL* y utilizando la función de columna SUM en la lista de selección de la selección completa.

El resultado de la consulta es como sigue:

| PIEZA | SUBPIEZA | Cant. total usada |
|-------|----------|-------------------|
| 01    | 02       | 2                 |
| 01    | 03       | 3                 |
| 01    | 04       | 4                 |
| 01    | 05       | 14                |
| 01    | 06       | 15                |
| 01    | 07       | 18                |
| 01    | 08       | 40                |
| 01    | 09       | 44                |
| 01    | 10       | 140               |
| 01    | 11       | 140               |
| 01    | 12       | 294               |
| 01    | 13       | 150               |
| 01    | 14       | 144               |

A la vista del resultado, considere la línea de la subpieza '06'. La cantidad total utilizada, con valor 15, deriva de la cantidad de 3 directamente para la pieza '01' y la cantidad de 6 para la pieza '02', que se necesita 2 veces en la pieza '01'.

---

## Ejemplo 3: Control de profundidad

Puede surgir la cuestión de qué es lo que ocurre cuando existen más niveles de piezas en la tabla de los que está interesado para su consulta. Es decir, cómo se escribe una consulta para responder a la pregunta "Cuáles son los dos primeros niveles de piezas necesarias para crear la pieza identificada como '01'?" Por cuestiones de claridad en el ejemplo, el nivel se incluye en el resultado.

```
WITH RPL (NIVEL, PIEZA, SUBPIEZA, CANTIDAD) AS
(
 SELECT 1, PIEZA.RAIZ SUBPIEZA.RAIZ, CANTIDAD.RAIZ
 FROM LISTA DE PIEZAS RAIZ
 WHERE PIEZA.RAIZ = '01'
 UNION ALL
 SELECT NIVEL+1.PADRE, PIEZA.HIJA, SUBPIEZA.HIJA, CANTIDAD.HIJA
 FROM RPL PADRE, LISTA DE PIEZAS HIJA
 WHERE SUBPIEZA.PADRE = PIEZA.HIJA
```

## Ejemplo de recurrencia: Lista de material

```
 AND NIVEL.PADRE < 2
)
SELECT PIEZA, NIVEL, SUBPIEZA, CANTIDAD
FROM RPL;
```

Esta consulta es similar al ejemplo 1. La columna *NIVEL* se ha introducido para contar los niveles desde la pieza original. En la selección completa de inicialización, el valor de la columna *NIVEL* se inicializa en 1. En la selección completa subsiguiente, el nivel padre se incrementa en 1. A continuación, para controlar el número de niveles del resultado, la segunda selección completa incluye la condición de que el nivel padre debe ser menor que 2. Esto garantiza que la segunda selección completa sólo procesará hijos en el segundo nivel.

El resultado de la consulta es como sigue:

| PIEZA | NIVEL | SUBPIEZA | CANTIDAD |
|-------|-------|----------|----------|
| 01    | 1     | 02       | 2        |
| 01    | 1     | 03       | 3        |
| 01    | 1     | 04       | 4        |
| 01    | 1     | 06       | 3        |
| 02    | 2     | 05       | 7        |
| 02    | 2     | 06       | 6        |
| 03    | 2     | 07       | 6        |
| 04    | 2     | 08       | 10       |
| 04    | 2     | 09       | 11       |
| 06    | 2     | 12       | 10       |
| 06    | 2     | 13       | 10       |

## Ejemplo de recurrencia: Lista de material

---

## Apéndice N. Tablas de excepciones

Las tablas de excepciones son tablas creadas por el usuario que imitan la definición de las tablas cuya comprobación se especifica utilizando SET INTEGRITY con la opción IMMEDIATE CHECKED. Se utilizan para almacenar copias de las filas que violan las restricciones de las tablas que se están comprobando.

Las tablas de excepciones que se utilizan con LOAD son idénticas a las utilizadas aquí. Por lo tanto, se pueden volver a utilizar durante la comprobación con la sentencia SET INTEGRITY.

---

### Reglas para crear una tabla de excepciones

Las reglas para crear una tabla de excepciones son las siguientes:

1. Las primeras “n” columnas de la tabla de excepciones son iguales que las columnas de la tabla que se está comprobando. Todos los atributos de columna inclusive el nombre, el tipo y la longitud deben ser idénticos.
2. Todas las columnas de la tabla de excepciones deben estar libres de cualquier restricción y desencadenante. Las restricciones incluyen la integridad de referencia, las restricciones de comprobación así como las restricciones de índice de unicidad que podrían causar errores en la inserción.
3. La columna “(n+1)” de la tabla de excepciones es una columna TIMESTAMP opcional. Esto sirve para identificar las invocaciones sucesivas de la comprobación que efectúa la sentencia SET INTEGRITY en la misma tabla, si las filas de la tabla de excepciones no se han suprimido antes mediante la emisión de la sentencia SET INTEGRITY para comprobar los datos.
4. La columna “(n+2)” debe ser de tipo CLOB(32K) o mayor. Esta columna es opcional pero se recomienda incluirla y se utilizará para proporcionar los nombres de las restricciones que violan los datos de la fila. Si no se proporciona esta columna (como pasaría si, por ejemplo, la tabla original tuviese el número máximo de columnas permitido), sólo se copia la fila en la que se ha detectado la violación de restricción.
5. La tabla de excepciones debe crearse con las columnas “(n+1)” y “(n+2)”.
6. No se impone ningún nombre en particular para las columnas adicionales anteriores. No obstante, debe seguirse exactamente la especificación del tipo.
7. No se permiten columnas adicionales.

8. Si la tabla original tiene columnas DATALINK, las columnas correspondientes de la tabla de excepciones deben especificar NO LINK CONTROL. Esto asegura que no se enlace un archivo cuando se inserte una fila (con columna DATALINK) ni se genere un símbolo de accesos cuando se seleccionen filas de la tabla de excepciones.
9. Si la tabla original tiene columnas generadas (incluida la propiedad IDENTITY), las columnas correspondientes de la tabla de excepciones no deben especificar la propiedad generada.
10. También debe tenerse en cuenta que los usuarios que invocan SET INTEGRITY para comprobar los datos deben tener el privilegio INSERT en las tablas de excepciones.

La información de la columna “mensaje” tiene la siguiente estructura:

Tabla 148. Estructura de la columna de mensajes de la tabla de excepciones

| Número de campo | Contenido                                                                                             | Tamaño                      | Comentarios                                                                                                                                                                                                             |
|-----------------|-------------------------------------------------------------------------------------------------------|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1               | Número de violaciones de restricción                                                                  | 5 caracteres                | Justificada por la derecha rellenada con '0'                                                                                                                                                                            |
| 2               | Tipo de la primera violación de restricción                                                           | 1 carácter                  | 'K' - Violación de restricción de comprobación<br>'F' - Violación de clave foránea<br>'G' - Violación de columna generada<br>'I' - Violación de índice de unicidad <sup>a</sup><br>'L' - Violación de carga de DATALINK |
| 3               | Longitud de restricción/columna <sup>b</sup> /ID índice <sup>c</sup> /DLVDESC <sup>d</sup>            | 5 caracteres                | Justificada por la derecha rellenada con '0'                                                                                                                                                                            |
| 4               | Nombre de restricción/Nombre de columna <sup>b</sup> /ID de índice <sup>c</sup> /DLVDESC <sup>d</sup> | longitud del campo anterior |                                                                                                                                                                                                                         |
| 5               | Separador                                                                                             | 3 caracteres                | <espacio><:><espacio>                                                                                                                                                                                                   |
| 6               | Tipo de la siguiente violación de restricción                                                         | 1 carácter                  | 'K' - Violación de restricción de comprobación<br>'F' - Violación de clave foránea<br>'G' - Violación de columna generada<br>'I' - Violación de índice de unicidad<br>'L' - Violación de carga de DATALINK              |



Tabla 148. Estructura de la columna de mensajes de la tabla de excepciones (continuación)

| Número de campo                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Contenido                                                     | Tamaño                      | Comentarios                                  |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------|-----------------------------|----------------------------------------------|
| 7                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Longitud de restricción/columna/ID de índice/ DLVDESC         | 5 caracteres                | Justificada por la derecha rellenada con '0' |
| 8                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Nombre de restricción/Nombre de columna/ID de índice/ DLVDESC | longitud del campo anterior |                                              |
| .....                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | .....                                                         | .....                       | Repita del Campo 5 al 8 para cada violación  |
| <ul style="list-style-type: none"> <li>• <sup>a</sup> No se producirán violaciones de índices de unicidad en la comprobación si se utiliza SET INTEGRITY. Sin embargo, se informará de esto, cuando se ejecute LOAD si se elige la opción FOR EXCEPTION. Por otra parte, LOAD no informará de las violaciones de restricción de comprobación, de columna generada ni de clave foránea ocurridas en las tablas de excepciones.</li> <li>• <sup>b</sup> Para recuperar la expresión de una columna generada a partir de las vistas de catálogo, utilice una sentencia de selección. Por ejemplo, si el campo 4 es MYSCHEMA.MYTABLE.GEN_1, entonces SELECT SUBSTR(TEXT, 1, 50) FROM SYSCAT.COLUMNS WHERE TABSCHEMA='MYSCHEMA' AND TABNAME='MYNAME' AND COLNAME='GEN_1'; devuelve los primeros 50 caracteres de la expresión, en el formato "AS (&lt;expresión&gt;)"</li> <li>• <sup>c</sup> Para recuperar un ID de índice a partir de las vistas de catálogo, utilice una sentencia de selección. Por ejemplo, si el campo 4 es 1234, entonces SELECT INDSHEMA, INDNAME FROM SYSCAT.INDEXES WHERE IID=1234.</li> <li>• <sup>d</sup>DLVDESC es un DESCriptor de Violación de carga de DATALINK, que se describe a continuación.</li> </ul> |                                                               |                             |                                              |

Tabla 149. DESCriptor de Violación de carga de DATALINK (DLVDESC)

| Número de campo | Contenido                                                     | Tamaño       | Comentarios                                        |
|-----------------|---------------------------------------------------------------|--------------|----------------------------------------------------|
| 1               | Número de columnas DATALINK de violación                      | 4 caracteres | Justificada por la derecha rellenada con '0'       |
| 2               | Número de columna DATALINK de la primera columna de violación | 4 caracteres | Justificada por la derecha rellenada con '0'       |
| 2               | Número de columna DATALINK de la segunda columna de violación | 4 caracteres | Justificada por la derecha rellenada con '0'       |
| .....           | .....                                                         | .....        | Se repite para cada número de columna de violación |

**Nota:**

- El número de columna DATALINK es COLNO en SYSCAT.COLUMNS para la tabla adecuada.

---

## Manejo de filas en las tablas de excepciones

La información de las tablas de excepciones se puede procesar de la forma que se desee. Las filas pueden utilizarse para corregir los datos y volver a insertar las filas en las tablas originales.

Si no hay ningún desencadenante INSERT en la tabla original, transfiera las filas corregidas emitiendo la sentencia INSERT con una subconsulta en la tabla de excepciones.

Si hay desencadenantes INSERT y desea completar la carga con las filas corregidas de las tablas de excepciones sin disparar los desencadenantes, se sugieren las siguientes maneras:

- Diseñe los desencadenantes INSERT para que se disparen dependiendo del valor de una columna definida explícitamente para esta finalidad.
- Descargue los datos de las tablas de excepciones y añádalos utilizando LOAD. En este caso si se vuelven a comprobar los datos, se debe tener en cuenta que en DB2 Versión 7 la comprobación de la violación de restricción no está confinada solamente a las filas añadidas.
- Guarde el texto del desencadenante de la tabla de catálogos relevante. Después elimine el desencadenante INSERT y utilice INSERT para transferir las filas corregidas de las tablas de excepciones. Finalmente vuelva a crear el desencadenante utilizando la información guardada.

En DB2 Versión 7, no se realiza una provisión explícita para evitar que se disparen los desencadenantes cuando se insertan filas de las tablas de excepciones.

Sólo se informará de una violación por fila para las violaciones de índices de unicidad.

Si hay valores con series largas o con tipos de datos LOB en la tabla, los valores no se insertarán en la tabla de excepciones en caso de violación de índice de unicidad.

---

## Consulta de las tablas de excepciones

La estructura de la columna de mensajes de una tabla de excepciones es una lista concatenada de nombres de restricciones, longitudes y delimitadores tal como se describe antes. Es posible que desee escribir una consulta sobre esta información.

Por ejemplo, supongamos que se escribe una consulta para obtener una lista de todas las violaciones, repitiendo cada fila con sólo el nombre de la restricción junto a ella. Supongamos que nuestra tabla original T1 tenía dos columnas C1 y C2. Supongamos también que la tabla de excepciones

correspondiente E1 tiene las columnas C1, C2 que pertenecen a las de T1 y MSGCOL como la columna de mensajes. La siguiente consulta (utilizando recurrencia) listará un nombre de restricción por fila (que pertenece a la fila para más de una violación):

```
WITH IV (C1, C2, MSGCOL, CONSTNAME, I, J) AS
 (SELECT C1, C2, MSGCOL,
 CHAR(SUBSTR(MSGCOL, 12,
 INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0)))),
 1,
 15+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0))
 FROM E1
UNION ALL
 SELECT C1, C2, MSGCOL,
 CHAR(SUBSTR(MSGCOL, J+6,
 INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0)))),
 I+1,
 J+9+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0))
 FROM IV
 WHERE I < INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,1,5)),5,0))
) SELECT C1, C2, CONSTNAME FROM IV;
```

Si deseamos que todas las filas que han violado una restricción en particular, ampliaríamos esta consulta de la siguiente manera:

```
WITH IV (C1, C2, MSGCOL, CONSTNAME, I, J) AS
 (SELECT C1, C2, MSGCOL,
 CHAR(SUBSTR(MSGCOL, 12,
 INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0)))),
 1,
 15+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0))
 FROM E1
UNION ALL
 SELECT C1, C2, MSGCOL,
 CHAR(SUBSTR(MSGCOL, J+6,
 INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0)))),
 I+1,
 J+9+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0))
 FROM IV
 WHERE I < INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,1,5)),5,0))
) SELECT C1, C2, CONSTNAME FROM IV WHERE CONSTNAME = 'nombrerestricción';
```

Para obtener todas las violaciones de restricciones de comprobación, se podría ejecutar lo siguiente:

```
WITH IV (C1, C2, MSGCOL, CONSTNAME, CONSTTYPE, I, J) AS
 (SELECT C1, C2, MSGCOL,
 CHAR(SUBSTR(MSGCOL, 12,
 INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0))),
 CHAR(SUBSTR(MSGCOL, 6, 1)),
 1,
 15+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0))
 FROM E1
UNION ALL
 SELECT C1, C2, MSGCOL,
```

```
CHAR(SUBSTR(MSGCOL, J+6,
 INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0)))),
CHAR(SUBSTR(MSGCOL, J, 1)),
I+1,
J+9+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0))
FROM IV
WHERE I < INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,1,5)),5,0))
) SELECT C1, C2, CONSTNAME FROM IV WHERE CONSTTYPE = 'K';
```

---

## Apéndice O. Consideraciones acerca de EUC de japonés y de chino tradicional

Extended Unix Code (EUC) para japonés y chino tradicional define un conjunto de normas de codificación que dan soporte a juegos de 1 a 4 caracteres. En algunos casos como, por ejemplo, el EUC para japonés (eucJP) y el EUC para chino tradicional (eucTW), puede codificarse un carácter utilizando más de dos bytes. La utilización de un esquema de codificación así tiene implicaciones cuando se utiliza como la página de códigos del servidor de bases de datos o como el cliente de base de datos. Las consideraciones clave implican lo siguiente:

- la expansión o contracción de series cuando se convierte entre las páginas de códigos EUC y páginas de códigos de doble byte
- el uso del Juego de caracteres universal-2 (UCS-2) como la página de códigos para los datos gráficos almacenados en un servidor de base de datos definido con las páginas de códigos eucJP (japonés) o eucTW (chino tradicional).

A excepción de estas consideraciones, la utilización de EUC es coherente con el soporte del juego de caracteres de doble byte (DBCS). En todo este manual (y otros), las referencias a *doble byte* se han cambiado por *múltiples bytes* para reflejar el soporte de las reglas de codificación que permiten las representaciones de caracteres que necesitan más de 2 bytes. Se incluyen aquí consideraciones detalladas para dar soporte al EUC para japonés y para chino tradicional, organizadas de la misma manera que el contenido de los capítulos de este manual. Deben tener en cuenta esta información las personas que utilicen SQL con un servidor de bases de dtos EUC o un cliente de base de datos EUC y debe utilizarse junto con la información sobre el desarrollo de aplicaciones del manual *Application Development Guide*

---

### Elementos del lenguaje

#### Caracteres

Cada carácter de múltiples bytes se considera una *letra* a excepción de un carácter blanco de doble byte que se considera un *carácter especial*.

#### Símbolos

Las letras alfabéticas en minúsculas de múltiples bytes no se transforman a mayúsculas. Esto difiere de la letras alfabéticas en minúsculas de un solo byte en los símbolos que generalmente se convierten a mayúsculas.

# Consideraciones acerca de EUC de japonés y de chino tradicional

## Identificadores

### Identificadores SQL

La conversión entre una página de códigos de doble byte y una página de códigos EUC puede dar como resultado la conversión de caracteres de doble byte en caracteres de múltiples bytes codificados con más de 2 bytes. Como resultado, los identificadores que quepan en la longitud máxima de la página de códigos de doble byte pueden exceder la longitud de la página de códigos EUC. La selección de identificadores para este tipo de entorno debe realizarse con cuidado para evitar la expansión más allá de la longitud máxima de identificador.

## Tipos de datos

### Series de caracteres

En una base de datos MBCS, las series de caracteres pueden contener una mezcla de caracteres del juego de caracteres de un solo byte (SBCS) y de juegos de caracteres de múltiples bytes (MBCS). Cuando se utilizan dichas series, las operaciones pueden proporcionar resultados distintos si se basan en caracteres (tratan los datos como caracteres) a si se basan en bytes (tratan los datos como bytes). Compruebe la descripción de la función o de la operación para determinar cómo se procesan las series mixtas.

### Series gráficas

Una serie gráfica se define como una secuencia de datos de caracteres de doble byte. Para permitir que los datos del EUC japonés o chino tradicional se almacenen en columnas gráficas, los caracteres del EUC se codifican en UCS-2. Los caracteres que no son de doble byte bajo todos los esquemas de codificación soportados (por ejemplo, PC o EBCDIC DBCS) no deben utilizarse en columnas gráficas. Los resultados de la utilización de otros caracteres de doble byte pueden ser que se reemplacen por caracteres de sustitución durante la conversión. La recuperación de dichos datos no devolverá el mismo valor que se ha entrado. Consulte el manual *Application Development Guide* para ver los detalles acerca del manejo de datos gráficos en variables del lenguaje principal.

## Asignaciones y comparaciones

### Asignaciones de series

La conversión de una serie se realiza antes de la asignación. En los casos que implican una página de códigos eucJP/eucTW y una página de códigos DBCS, una serie de caracteres puede convertirse en más larga (DBCS a eucJP/eucTW) o más corta (eucJP/eucTW a DBCS). Puede dar como resultado errores en la asignación de almacenamiento y el truncamiento en asignaciones de recuperación. Cuando el error en la asignación de almacenamiento es debido a la expansión durante la conversión, se devuelve SQLSTATE 22524 en lugar de SQLSTATE 22001.

## Consideraciones acerca de EUC de japonés y de chino tradicional

De igual forma, las asignaciones que implican series gráficas pueden dar como resultado la conversión de un carácter de doble byte codificado UCS-2 a un carácter de sustitución en una página de códigos PC o EBCDIC DBCS para los caracteres que no tienen un carácter de doble byte correspondiente. Las asignaciones que sustituyen los caracteres por caracteres de sustitución lo indicarán estableciendo el campo SQLWARN10 de la SQLCA en 'W'.

En los casos de truncamiento durante la asignación de recuperación que implica series de caracteres de doble byte, el punto de truncamiento puede formar parte de un carácter de múltiples bytes. En este caso, cada byte del fragmento del carácter se sustituye por un blanco de un solo byte. Esto significa que más de un blanco de un solo byte puede aparecer al final de una serie de caracteres truncada.

### Comparaciones de series

Las comparaciones de series se realizan sobre la base de los bytes. Las series de caracteres también utilizan el orden de clasificación definido para la base de datos. Las series gráficas no utilizan el orden de clasificación y, en una base de datos eucJP o eucTW, se codifican utilizando UCS-2. Por lo tanto, la comparación de dos series de caracteres mixtos puede dar un resultado diferente de la comparación de dos series gráficas aunque contengan los mismos caracteres. De igual forma, el orden de clasificación resultante de una columna de caracteres mixtos y una columna gráfica puede ser diferente.

### Reglas para los tipos de datos del resultado

El tipo de datos resultante para series de caracteres no se ve afectado por la posible expansión de la serie. Por ejemplo, una unión de dos operandos CHAR seguirá siendo CHAR. No obstante, si uno de los operandos de la serie de caracteres se va a convertir de manera que la expansión máxima convierta al atributo de longitud en el más grande de los dos operandos, entonces el atributo de longitud de la serie de caracteres resultante se verá afectado. Por ejemplo, considere las expresiones del resultado de una expresión CASE que tengan tipos de datos de VARCHAR(100) y VARCHAR(120). Suponga que la expresión VARCHAR(100) sea una variable del lenguaje principal de serie mixta (que pueda necesitar conversión) y la expresión VARCHAR(120) sea una columna en la base de datos eucJP. El tipo de datos resultante es VARCHAR(200) ya que VARCHAR(100) se dobla para permitir una posible conversión. El mismo escenario sin la implicación de una base de datos eucJP ni eucTW tendría un tipo del resultado de VARCHAR(120).

Tenga en cuenta que doblar la longitud de la variable del lenguaje principal se basa en el hecho de que el servidor de bases de datos es EUC japonés o EUC chino tradicional. Incluso si el cliente también es eucJP o eucTW, se sigue doblando. Esto permite que pueda utilizarse el mismo paquete de aplicación por clientes de doble byte o de múltiples bytes.

## Consideraciones acerca de EUC de japonés y de chino tradicional

### Reglas para las conversiones de series

Los tipos de operaciones listadas en la sección correspondiente del manual Consulta de SQL pueden convertir los operandos a la página de códigos de la aplicación o de la base de datos.

Si dichas operaciones se realizan en un entorno de páginas de códigos mixtas que incluye EUC del japonés o chino tradicional, puede producirse expansión o contracción de los operandos de series de caracteres mixtos. Por lo tanto, el tipo de datos resultante tiene un atributo de longitud que acomoda la expansión máxima, si es posible. En los casos en que existen restricciones en el atributo de longitud del tipo de datos, se utiliza la longitud máxima permitida para el tipo de datos. Por ejemplo en un entorno en que el crecimiento máximo es el doble, una variable del lenguaje principal VARCHAR(200) se trata como si fuese VARCHAR(400), pero la variable del lenguaje principal CHAR(200) se trata como si fuese CHAR(254). Puede producirse un error en tiempo de ejecución cuando se realiza una conversión si la serie convertida va a exceder la longitud máxima del tipo de datos. Por ejemplo, la unión de CHAR(200) y CHAR(10) tendría un tipo del resultado de CHAR(254). Cuando se convierte el valor del lado izquierdo de la UNION, si se necesitan más de 254 caracteres, se produce un error.

En algunos casos, si se permite el crecimiento máximo para la conversión provocará que el atributo de longitud exceda el límite. Por ejemplo, UNION sólo permite columnas de un máximo de 254 bytes. Por lo tanto, una consulta con una unión que incluye una variable del lenguaje principal en la lista de columnas (llamémosla :hv1) que es una serie de caracteres mixta DBCS definida como una serie de caracteres de longitud variable de 128 bytes de longitud, establecerá el tipo de datos en VARCHAR(256) dando como resultado un error al preparar la consulta, aunque la consulta de la aplicación no parezca tener ninguna columna mayor que 254. En una situación en que no sea probable que la serie real vaya a producir una expansión más allá de 254 bytes se puede utilizar lo siguiente para preparar la sentencia.

```
SELECT CAST(:hv1 CONCAT ' AS VARCHAR(254)), C2 FROM T1
UNION
SELECT C1, C2 FROM T2
```

La concatenación de la serie nula con la variable del lenguaje principal forzará a que se produzca la conversión antes de que se ejecute la función cast. Esta consulta se puede preparar en DBCS para el entorno eucJP/eucTW aunque puede producirse un error de truncamiento en tiempo de ejecución.

Esta técnica (serie nula concatenada con cast) se puede utilizar para manejar el límite similar de 257 bytes para SELECT DISTINCT o para utilizar la columna en las cláusulas ORDER BY o GROUP BY.



## Constantes

### Constantes gráficas de tipo serie

El cliente EUC japonés o chino tradicional, puede contener caracteres de un solo byte o de múltiples bytes (como una serie de caracteres mixta). La serie no debe contener más de 2.000 caracteres. Se recomienda que sólo se utilicen en constantes gráficas los caracteres que se convierten a caracteres de doble byte en todas las páginas de códigos de doble byte PC y EBCDIC relacionadas. Una constante de serie gráfica de una sentencia de SQL se convierte de la página de códigos cliente a la codificación de doble byte en el servidor de la base de datos. Para un servidor de UC japonés o chino tradicional, la constante se convierte a UCS-2, la codificación de doble byte utilizada para series gráficas. Para un servidor de doble byte, la constante se convierte de la página de códigos cliente a la página de códigos DBCS del servidor.

## Funciones

El diseño de las funciones definidas por el usuario debe tomar en consideración el efecto del soporte de EUC japonés o chino tradicional en los tipos de datos de parámetros. Una parte de la resolución de la función considera los tipos de datos de los argumentos para una llamada a función. Los argumentos de la serie de caracteres mixtos que implican un cliente EUC japonés o chino tradicional pueden necesitar bytes adicionales para especificar el argumento. Esto puede necesitar que el tipo de datos cambie para permitir la longitud incrementada. Por ejemplo, pueden utilizarse 4001 bytes para representar una serie de caracteres en la aplicación (LONG VARCHAR) que quepa en una serie VARCHAR(4000) en el servidor. Si no se incluye una signature de función que permita que el argumento sea LONG VARCHAR, la resolución de la función no podrá buscar una función.

Existen algunas funciones que no permiten las series largas por varias razones. La utilización de los argumentos LONG VARCHAR o CLOB con dichas funciones no será satisfactoria. Por ejemplo, LONG VARCHAR como segundo argumento de la función POSSTR incorporada, hará fallar la resolución de función (SQLSTATE 42884).

## Expresiones

### Con el operador de concatenación

La expansión potencial de uno de los operandos de la concatenación puede hacer que el tipo de datos y la longitud de los operandos concatenados cambien cuando se incluye un servidor de EUC del japonés o chino tradicional en un entorno. Por ejemplo, con un servidor EUC en el que el valor de una variable del lenguaje principal puede doblar su longitud.

```
CHAR200 CONCAT :char50
```

## Consideraciones acerca de EUC de japonés y de chino tradicional

La columna *CHAR200* es de tipo CHAR(200). La variable del lenguaje principal *char50* se define como CHAR(50). El tipo del resultado para esta operación de concatenación normalmente sería CHAR(250). No obstante, dado un servidor de bases de datos eucJP o eucTW, se asume que la serie puede expandir su longitud al doble. Por lo tanto, *char50* se trata como CHAR(100) y el tipo resultante es VARCHAR(300). Tenga en cuenta que aunque el resultado sea VARCHAR, siempre tendrá 300 bytes de datos incluyendo blancos de cola. Si no se desean los blancos de cola adicionales, defina la variable del lenguaje principal como VARCHAR(50) en lugar de CHAR(50).

### Predicados

#### Predicado LIKE

Para un predicado LIKE que implique series de caracteres mixtas en una base de datos EUC:

- el subrayado de un solo byte representa cualquier carácter de un solo byte
- el signo del tanto por ciento de un solo byte representa una serie de cero o varios caracteres (caracteres de un solo byte o de múltiples bytes).
- el subrayado de doble byte representa cualquier carácter de múltiples bytes
- el signo de tanto por ciento de doble byte representa una serie de cero o varios caracteres (caracteres de un solo byte o de múltiples bytes)

El carácter de escape debe ser un carácter de un solo byte o un carácter de doble byte.

Tenga en cuenta que la utilización de un carácter de subrayado puede producir resultados diferentes dependiendo de la página de códigos de la operación LIKE. Por ejemplo, los caracteres Katakana de EUC del japonés son caracteres de múltiples bytes (CS2) pero en la página de códigos DBCS del japonés son caracteres de un solo byte. Una consulta con el subrayado de un solo byte en la *expresión-patrón* devolvería ocurrencias del carácter Katakana en la posición del subrayado de un servidor DBCS del japonés. Sin embargo, no se devolvería las mismas filas de la tabla equivalente en un servidor EUC del japonés, ya que los caracteres Katakana sólo coinciden con un subrayado de doble byte.

Para un predicado LIKE que implique series gráficas en una base de datos EUC:

- el carácter utilizado para el subrayado y signo del tanto por ciento debe correlacionarse con el carácter del subrayado y de tanto por ciento respectivamente
- el subrayado representa cualquiera carácter UCS-2
- el tanto por ciento representa una serie de cero o varios caracteres UCS-2.

### Funciones

#### LENGTH

El proceso de esta función no es diferente para las series de caracteres mixtas en un entorno EUC. El valor devuelto es la longitud de la serie en la página de códigos del argumento. Cuando se utilice esta función para determinar la longitud de un valor, debe tenerse especial cuidado en cómo se utiliza la longitud. En especial para las constantes de series mixtas ya que la longitud se da en bytes, no en caracteres. Por ejemplo, la longitud de una columna de serie mixta en una base de datos DBCS devuelta por la función LENGTH puede ser menor que la longitud del valor recuperado de dicha columna en un cliente eucJP o eucTW debido a la conversión de algunos caracteres DBCS para caracteres eucJP o eucTW de múltiples bytes.

#### SUBSTR

La función SUBSTR funciona en series de caracteres mixtas basándose en los bytes. Por lo tanto, la serie resultante puede incluir fragmentos de caracteres de múltiples bytes al principio o al final de la serie resultado. No se proporciona ningún proceso para detectar o procesar fragmentos de caracteres.

#### TRANSLATE

La función TRANSLATE da soporte a series de caracteres mixtos que incluyen caracteres de múltiples bytes. Los caracteres correspondientes de la *exp-a-serie* y de la *exp-de-serie* deben tener el mismo número de bytes y no pueden finalizar con parte de un carácter de múltiples bytes.

La *exp-car-relleno* debe resultar en un carácter de un solo byte cuando la *exp-serie-car* sea una serie de caracteres. Puesto que se lleva a cabo TRANSLATE en la página de códigos de la *exp-serie-car*, la *exp-car-relleno* puede convertirse de un carácter de múltiples bytes a un carácter de un solo byte.

No se convertirán estos bytes de una *exp-serie-car* que finaliza con parte de un carácter de múltiples bytes.

#### VARGRAPHIC

La función VARGRAPHIC en un operando de serie de caracteres de una página de códigos EUC del japonés o del chino tradicional devuelve una serie gráfica en la página de códigos UCS-2.

- Los caracteres de un solo byte se convierten primero a sus caracteres de doble byte correspondientes en el conjunto de códigos al que pertenecen (eucJP o eucTW). Después, se convierten a la representación UCS-2 correspondiente. Si no hay ninguna representación de doble byte, el carácter se convierte al carácter de sustitución de doble byte definido para dicho conjunto de códigos antes de convertirse a la representación UCS-2.

## Consideraciones acerca de EUC de japonés y de chino tradicional

- Los caracteres de eucJP que son Katakana (eucJP CS2) son realmente caracteres de un solo byte en algunos esquemas de codificación. Por lo tanto, se convierten a los caracteres de doble byte correspondientes en eucJP o al carácter de sustitución de doble byte antes de convertirse a UCS-2.
- Los caracteres de múltiples bytes se convierten a sus representaciones UCS-2.

---

## Sentencias

### CONNECT

El proceso de una sentencia CONNECT satisfactoria devuelve información en la SQLCA que es importante cuando existe la posibilidad de que las aplicaciones procesen los datos en un entorno que incluya una página de códigos EUC del japonés o del chino tradicional en el cliente o en el servidor. El campo *SQLERRD(1)* proporciona la expansión máxima de una serie de caracteres mixta cuando se convierte de la página de códigos de la aplicación a la página de códigos de la base de datos. El campo *SQLERRD(2)* proporciona la expansión máxima de una serie de caracteres mixta cuando se convierte de la página de códigos de la base de datos a la página de códigos de la aplicación. El valor es positivo si se ha podido producir la expansión y negativo si se ha podido producir la contracción. Si el valor es negativo, el valor siempre es -1 ya que en el peor de los casos es que no se produce ninguna contracción y que se necesita toda la longitud de la serie después de la conversión. Los valores positivos pueden ser como máximo 2, que significa que en el peor de los casos, puede necesitarse el doble de la longitud de la serie para la serie de caracteres después de la conversión.

La página de códigos para el servidor de aplicaciones y la aplicación cliente también está disponible en el campo *SQLERRMC* de la SQLCA.

### PREPARE

Los tipos de datos determinados para los marcadores de parámetros sin tipo (tal como se describen en la Tabla 30 en la página 1102) no se cambian en un entorno que incluya EUC del japonés o del chino tradicional. Como resultado, puede ser necesario en algunos casos utilizar marcadores de parámetros con tipo para proporcionar la longitud suficiente para las series de caracteres mixtas en eucJP o eucTW. Por ejemplo, considere una inserción en una columna CHAR(10). Preparación de la sentencia:

```
INSERT INTO T1 (CH10) VALUES (?)
```

dará como resultado un tipo de datos de CHAR(10) para el marcador de parámetros. Si el cliente era eucJP o eucTW, pueden ser necesarios más de 10 bytes para representar la serie que se ha de insertar pero la misma serie en la página de códigos DBCS de la base de datos no tiene más de 10 bytes. En este

## Consideraciones acerca de EUC de japonés y de chino tradicional

caso, la sentencia que se ha de preparar debe incluir un marcador de parámetro con tipo con una longitud mayor que 10. Por lo tanto, la preparación de la sentencia:

```
INSERT INTO T1 (CH10) VALUES (CAST(? AS VARCHAR(20))
```

dará como resultado un tipo de datos de VARCHAR(20) para el marcador de parámetros.



---

## Apéndice P. Especificaciones BNF para los enlaces de datos

Un valor DATALINK es un valor encapsulado que contiene una referencia lógica de la base de datos a un archivo almacenado fuera de la base de datos.

El atributo de ubicación de datos de este valor encapsulado es una referencia lógica expresada en forma de URL (Uniform Resource Locator, localizador uniforme de recursos). El valor de este atributo sigue la sintaxis aplicable a los URL, tal como indica la siguiente especificación BNF<sup>120</sup>, basada en el RFC 1738 : Uniform Resource Locators (URL), T. Berners-Lee, L. Masinter, M. McCahill, December 1994

Se utilizan los convenios siguientes en la especificación BNF:

- "|" designa alternativas
- los corchetes [ ] delimitan elementos opcionales o repetidos
- los literales aparecen entre comillas dobles ""
- los elementos pueden ir precedidos por [n]\* para representar n o más repeticiones del elemento que sigue; si no se especifica n, el valor por omisión es 0

Especificación BNF para enlaces de datos (DATALINK):

### URL

```
url = httpurl | fileurl | uncurl | dfsurl | emptyurl
```

### HTTP

```
httpurl = "http://" hostport ["/" hpath]
hpath = hsegment *["/" hsegment]
hsegment = *[uchar | ";" | ":" | "@" | "&" | "="]
```

Observe que se ha eliminado el elemento de búsqueda existente en la especificación BNF original definida en el RFC1738, pues no es una parte esencial de la referencia a archivo y no tiene ninguna utilidad en el contexto de los enlaces de datos.

### FILE

```
fileurl = "file://" host "/" fpath
fpath = fsegment *["/" fsegment]
fsegment = *[uchar | "?" | ":" | "@" | "&" | "="]
```

---

120. BNF es un acrónimo de "Backus Naur Form" - una notación formal para describir la sintaxis de un lenguaje determinado

Observe que host no es opcional y la serie "localhost" no tiene ningún significado especial, a diferencia de RFC1738. Esto evita interpretaciones confusas de "localhost" en las configuraciones cliente/servidor y de EEE.

### UNC

```
uncurl = "unc:\\\" hostname "\\\" sharename "\\\" uncpath
sharename = *uchar
uncpath = fsegment *["\\\" fsegment]
```

Se da soporte al convenio UNC habitual para nombres en NT. Esto no es un modelo estándar en RFC1738.

### DFS

```
dfsurl = "dfs://.../" cellname "/" fpath
cellname = hostname
```

Da soporte al convenio DFS para nombres. Esto no es un modelo estándar en RFC1738.

### EMPTYURL

```
emptyurl = ""
hostport = host [":" port]
host = hostname | hostnumber
hostname = *[domainlabel "."] toplabel
domainlabel = alphadigit | alphadigit *[alphadigit | "-"] alphadigit
toplabel = alpha | alpha *[alphadigit | "-"] alphadigit
alphadigit = alpha | digit
hostnumber = digits "." digits "." digits "." digits
port = digits
```

Los URL vacíos (de longitud cero) también pueden utilizarse para valores de tipo DATALINK. Son útiles para actualizar columnas DATALINK cuando se notifican excepciones de reconciliación e intervienen columnas DATALINK que no pueden contener nulos. Se utiliza un URL de longitud cero para actualizar la columna y provocar la desconexión

### Definiciones varias

```
lowalpha = "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" |
 "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" |
 "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" |
 "y" | "z"
hialpha = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" |
 "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" |
 "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" |
 "Y" | "Z"
alpha = lowalpha | hialpha
digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
 "8" | "9"
safe = "$" | "_" | "-" | "." | "+"
extra = "!" | "*" | "~" | "(" | ")" | ","
hex = digit | "A" | "B" | "C" | "D" | "E" | "F" |
 "a" | "b" | "c" | "d" | "e" | "f"
```



```
escape = "%" hex hex
unreserved = alpha | digit | safe | extra
uchar = unreserved | escape
digits = 1*digit
```

Los caracteres en blanco iniciales y de cola son eliminados por DB2 durante el análisis sintáctico. Además, los nombres de esquema ('HTTP', 'FILE', 'UNC', 'DFS') y host no son sensibles a las mayúsculas/minúsculas y se guardan siempre en mayúsculas en la base de datos.



---

## Apéndice Q. Glosario

### A

**acceso DRDA.** Método abierto de acceso a datos distribuidos mediante el cual se puede conectar a otro servidor de bases de datos (por ubicación), utilizando una sentencia de SQL, para ejecutar paquetes que se han enlazado anteriormente en dicha ubicación. Se utiliza la sentencia CONNECT de SQL o una sentencia de SQL de nombre de tres partes para identificar el servidor. Véase también “acceso por protocolo privado”.

**acceso por protocolo privado.** Método de acceso a datos distribuidos mediante el cual se puede dirigir una consulta hacia otro sistema DB2. Véase también “acceso DRDA”.

**acción desencadenada.** (1) Acción que se ejecuta cuando se produce el suceso desencadenante. (2) Lógica de SQL que se ejecuta cuando se activa un desencadenante. La acción desencadenada consta de una condición opcional de acción desencadenada y un conjunto de sentencias de SQL desencadenadas que sólo se ejecutan si el resultado de evaluar la condición es verdadero.

**acción de tarea.** En el Centro de tareas, acción que se realiza basándose en una condición que está relacionada con la ejecución de una tarea determinada. Por ejemplo, las acciones de tarea pueden ser: “Si la Tarea A se realiza satisfactoriamente, ejecutar la Tarea B,” y “Si la Tarea Z falla, inhabilitar la planificación de la Tarea Y.” Véase también “tarea” en la página 1617 y “paso” en la página 1590.

**activación del desencadenante.** Proceso que se produce cuando se ejecuta el suceso desencadenante indicado en una definición de desencadenante. La activación del desencadenante consiste en la evaluación de la condición de acción desencadenada y la ejecución condicional de las sentencias de SQL desencadenadas.

**actualización ascendente.** Proceso de actualización de los datos de una base de datos o un espacio de tablas restaurado mediante la aplicación de los cambios registrados en los archivos de anotaciones cronológicas de base de datos. Véase “recuperación en avance” en la página 1600.

**actualización asíncrona continua.** Proceso en el que todos los cambios efectuados en la fuente se registran y aplican a los datos de destino existentes después de confirmarlos en la tabla base. Véase también “actualización asíncrona de proceso por lotes”.

**actualización asíncrona de proceso por lotes.** Proceso en el que todos los cambios efectuados en la fuente se registran y aplican a los datos de destino existentes a intervalos especificados. Véase también “actualización asíncrona continua”.

**actualización de múltiples ubicaciones.** En DB2 Universal Database, proceso de bases de datos relacionales distribuidas en el que los datos se actualizan en más de una ubicación dentro de una unidad de trabajo individual.

**actualización de zona activa.** Serie de actualizaciones repetidas realizadas en las mismas filas durante un corto periodo de tiempo.

**actualización foránea.** Actualización que se ha efectuado en una tabla destino y se ha reproducido en la tabla local.

## Glosario

**actualización local.** Actualización en la tabla base, no en la duplicación.

**administrador de bases de datos (DBA).** Persona responsable del diseño, desarrollo, operación, seguridad, mantenimiento y utilización de una base de datos.

**administrador de duplicación.** Usuario responsable de definir suscripciones y fuentes de duplicación. Este usuario también puede ejecutar los programas Capture y Apply.

**administrador del sistema.** En un sistema informático, persona que diseña, controla y gestiona la utilización del sistema.

**ADSM.** ADSTAR Distributed Storage Manager. Véase “Tivoli Storage Manager” en la página 1619.

**agente.** (1) Proceso separado o hebra lleva a cabo todas las peticiones efectuadas a DB2 por una aplicación cliente determinada. (2) En DB2 Universal Database para OS/390, estructura que asocia todos los procesos que están implicados en una unidad de trabajo. Véase también “agente del sistema”, “agente de coordinación” y “agente aliado”.

**agente aliado.** Sinónimo de “hebra aliada” en la página 1567.

**agente de coordinación.** Agente que se inicia cuando el gestor de bases de datos recibe una petición de una aplicación. El agente permanece asociado a la aplicación mientras ésta se utiliza. Este agente coordina los subagentes que trabajan para la aplicación. Véase “agente”. Véase también “subagente” en la página 1611.

**agente de depósito.** En Centro de depósito de datos, proceso en tiempo de ejecución que gestiona el movimiento y la transformación de los datos.

**agente del sistema.** Petición de trabajo que DB2 Universal Database para OS/390 crea internamente, como por ejemplo un proceso de captación previa, escrituras diferidas y tareas de servicio. Véase “agente”.

**agente subordinado.** Véase “subagente” en la página 1611.

**agrupación de almacenamientos intermedios.** Área de memoria en la que se leen, modifican y se mantienen páginas de la base de datos durante el proceso.

**agrupación de almacenamientos intermedios de grupo (GBP).** Estructura de antememoria de recurso de acoplamiento utilizada por un grupo de compartimiento de datos para almacenar datos en antememoria y para asegurar que los datos son coherentes para todos los miembros. Véase también “estructura de antememoria” en la página 1561.

**agrupación de almacenamientos intermedios de grupo primario.** En una agrupación de almacenamientos intermedios de grupo duplicado, estructura de DB2 Universal Database para OS/390 que se utiliza para mantener la coherencia de los datos de la antememoria. Esta estructura se utiliza para el registro de páginas y la invalidación cruzada. El equivalente de OS/390 es la estructura *antigua*. Véase también “agrupación de almacenamientos intermedios de grupo secundario”.

**agrupación de almacenamientos intermedios de grupo secundario.** En una agrupación de almacenamientos intermedios de grupo duplicado, en el entorno DB2 Universal Database para OS/390, estructura que se utiliza para copiar las páginas modificadas que se escriben en la agrupación de almacenamientos intermedios de grupo primario. No se produce ningún registro de página ni

invalidación cruzada si se utiliza la agrupación de almacenamientos intermedios de grupo secundario. En OS/390, la estructura equivalente es *new*. Véase también “agrupación de almacenamientos intermedios de grupo primario” en la página 1526.

**agrupación de conexiones.** Cuando una aplicación solicita la desconexión del sistema principal, DB2 Connect elimina la conexión de entrada con la aplicación, pero conserva la conexión de salida al sistema principal en una agrupación. Cuando una aplicación nueva solicita una conexión, DB2 Connect utiliza una que se encuentre en la agrupación existente. La utilización de la conexión ya presente reduce el tiempo de conexión total, así como el alto coste de conexión de CPU en el sistema principal.

**agrupación de identificadores de registro (RID).** En DB2 Universal Database para OS/390, área de almacenamiento principal por encima de la línea de los 16 MB que está reservada para clasificar identificadores de registro durante el proceso de captación previa de lista.

**agrupación de Identificadores R.** Véase “agrupación de identificadores de registro”.

**agrupación EDM (electronic data management - gestión de datos electrónica).** En DB2 Universal Database para OS/390, agrupación de almacenamiento principal que se utiliza para descriptores de bases de datos, planes de aplicación, antememoria de autorización, paquetes de aplicación y colocación dinámica de sentencias en antememoria.

**agrupación en bloques.** Opción que se especifica cuando se enlaza una aplicación. Permite que el subsistema de comunicaciones ponga en antememoria varias filas de información para que cada sentencia FETCH no necesite la transmisión de una sola fila para cada petición enviada en la red. Véase también “captación de bloque” en la página 1536.

**alerta.** Acción, como, por ejemplo, un pitido o aviso, que se genera cuando una variable de rendimiento excede su valor umbral de aviso o alarma o llega por debajo de dicho umbral. Por ejemplo, la información sobre alertas se anota cronológicamente de forma automática en el Centro de alertas y en la página de Alertas del cuaderno de Diario.

**almacenamiento en antememoria.** Proceso mediante el cual se almacenan en memoria resultados de una petición que se utilizan con frecuencia para su rápida recuperación, hasta que llega el momento de renovar la información. En DB2 Universal Database existen varias formas de antememoria. Por ejemplo, almacenamiento en antememoria de directorio, almacenamiento en antememoria de paquete, almacenamiento en antememoria de sistema de archivos y almacenamiento en antememoria de LDAP.

**ámbito de grupo.** Véase 1578.

**ámbito del miembro.** Véase 1578.

**ámbito de mandato.** En DB2 Universal Database para OS/390, ámbito de actuación de un mandato en un grupo de compartimiento de datos. Si un mandato tiene *ámbito de miembro*, el mandato sólo muestra información de ese miembro o afecta solamente a recursos no compartidos que pertenecen localmente a dicho miembro. Si un mandato tiene *ámbito de grupo*, dicho mandato visualiza información de todos los miembros, afecta a los recursos no compartidos que son propiedad local de todos los miembros, visualiza información sobre los recursos que se pueden compartir o afecta a los recursos que se pueden compartir.

**American National Standard Code for Information Interchange (ASCII).** Véase “ASCII” en la página 1531.

## Glosario

**ampliación de hoja de cálculo.** En el Kit de iniciación de OLAP, software que se fusiona con Microsoft Excel y Lotus 1-2-3 para permitir el análisis multidimensional de datos. La biblioteca de software aparece como un elemento adicional de menú para la hoja de cálculo y proporciona funciones de análisis multidimensional tales como conectar, ampliar y calcular.

**análisis multidimensional.** Proceso de valoración y evaluación de una empresa a más de un nivel.

**analizador de sucesos.** Objeto de la base de datos que proporciona información sobre los sucesos de la base de datos que han tenido lugar. Un analizador de sucesos se utiliza junto con el archivo supervisor de sucesos para determinar y registrar información sobre el rendimiento.

**anotación cronológica.** (1) Archivo utilizado para registrar los cambios efectuados en un sistema. (2) Conjunto de registros que describen los sucesos que se producen durante la ejecución de DB2 Universal Database para OS/390 y que indican la secuencia de los sucesos. La información registrada de este modo se utiliza para la recuperación si se produce una anomalía durante la ejecución de DB2 Universal Database para OS/390. (3) Véase “anotación cronológica de base de datos”.

**anotación cronológica activa.** (1) Archivos de anotaciones cronológicas primario y secundario que actualmente son necesarios para la recuperación y la retroacción. (2) Parte de las anotaciones cronológicas de DB2 Universal Database para OS/390 donde se graban los registros de anotaciones a medida que se generan. La anotación cronológica activa contiene siempre los registros de anotación más recientes. Véase también “anotación cronológica de archivo”.

**anotación cronológica circular.** Archivo de anotaciones de una base de datos en el que los registros se sobregaban cuando ya no son necesarios para una base de datos activa. En consecuencia, si se produce un error, los datos perdidos no se pueden restaurar durante la recuperación ascendente. Véase también “anotación cronológica de base de datos” y “anotación cronológica de archivo”.

**anotación cronológica de archivo.** (1) Conjunto de archivos de anotaciones cronológicas que están cerrados y ya no se necesitan para el proceso normal. Estos archivos se conservan para utilizarlos en la recuperación ascendente. (2) Parte del archivo de anotaciones de DB2 Universal Database para OS/390 que contiene registros que se copian del archivo de anotaciones activo. La anotación cronológica de archivo contiene registros que son más antiguos y ya no caben en la anotación cronológica activa. Véase también “anotación cronológica activa”.

**anotación cronológica de base de datos.** Conjunto de archivos de anotaciones cronológicas primarios y secundarios formados por registros donde se anotan todos los cambios efectuados en una base de datos. El archivo de anotaciones cronológicas de la base de datos se utiliza para retrotraer cambios para unidades de trabajo que no están confirmadas y para recuperar el estado de coherencia de una base de datos.

**anotación cronológica primaria.** Conjunto de uno o más archivos de anotaciones cronológicas que se utilizan para registrar los cambios efectuados en una base de datos. El espacio de almacenamiento de estos archivos se asigna con antelación. Véase también “anotación cronológica secundaria” en la página 1529.

**anotación cronológica recuperable.** Anotación cronológica de base de datos en la que se conservan todos los registros de anotaciones para que, en caso de producirse una anomalía, los datos perdidos puedan recuperarse durante la recuperación ascendente. Véase también “anotación cronológica circular”.

**anotación cronológica secundaria.** Conjunto de uno o más archivos de anotaciones cronológicas utilizados para registrar los cambios efectuados en una base de datos. El espacio de almacenamiento para estos archivos se asigna según sea necesario cuando el archivo de anotaciones primario está lleno. Véase también “anotación cronológica primaria” en la página 1528.

**anotaciones cronológicas de servicio de primera anomalía.** Archivo (db2diag.log) que contiene mensajes de diagnóstico, datos de diagnóstico, información de alertas e información de vuelcos relacionada acerca de las operaciones de DB2. Este archivo lo utilizan los administradores de bases de datos para la resolución de problemas y la recuperación.

**antememoria.** Almacenamiento intermedio que contiene instrucciones y datos a los que se accede con frecuencia; se utiliza para reducir el tiempo de acceso.

**antiunión.** Conjunto de respuestas en el que las filas devueltas no cumplen la condición del predicado de unión. Véase “unión” en la página 1622.

**anulable.** Condición en la que puede estar ausente el valor de una columna, parámetro de función o de un resultado. Por ejemplo, el campo utilizado para la inicial del segundo apellido de una persona no necesita un valor y se considera anulable.

**APF.** Véase “recurso de programa autorizado” en la página 1601.

**API.** Véase “interfaz de programación de aplicación” en la página 1573.

**aplicación.** Programa o conjunto de programas que realiza una tarea; por ejemplo, aplicaciones de proceso de texto, gestión de inventario y cálculo de nóminas.

**apodo.** (1) Identificador que un servidor federado utiliza para hacer referencia a una fuente de datos, una tabla o una vista. (2) Nombre que se define en una base de datos de DB2 DataJoiner para representar un objeto físico de base de datos (tal como una tabla o procedimiento almacenado) en una base de datos no IBM.

**APPC.** Véase “Comunicación Avanzada Programa a Programa” en la página 1541.

**APPL.** Sentencia de definición de red VTAM que se utiliza para definir DB2 Universal Database para OS/390 en VTAM como un programa de aplicación que utiliza protocolos SNA LU 6.2.

**APPN.** Véase “Red avanzada de igual a igual” en la página 1601.

**archivo de anotaciones cronológicas.** Registro utilizado para supervisar la actividad de una base de datos. Los archivos de anotaciones cronológicas son esenciales para el proceso de copia de seguridad y recuperación.

**archivo de anotaciones de recuperación.** Véase “anotación cronológica de base de datos” en la página 1528.

**archivo de configuración.** Conjunto de valores de parámetros por omisión, por ejemplo los recursos asignados a los productos DB2 y a bases de datos individuales, y el nivel de diagnóstico. Existen dos tipos de archivos de configuración: el archivo de configuración de gestor de bases de datos para cada instancia DB2 y el archivo de configuración de base de datos para cada base de datos individual.

**archivo de enlace lógico.** Archivo producido por el precompilador cuando se utiliza el mandato PRECOMPILE o la API respectiva con la opción BINDFILE.

## Glosario

**archivo de índices.** Archivo que contiene información de indexado utilizada por el Video Extender para buscar una *toma de imagen* o fotograma individual en un segmento de vídeo.

**archivo de registro de auditoría.** Ubicación de los registros de auditoría generados desde el recurso de auditoría.

**archivo de respuestas.** Archivo ASCII que se puede personalizar con los datos de instalación y configuración que automatizarán una instalación. Los datos de instalación y configuración tienen que entrarse durante una instalación interactiva, pero con un archivo de respuestas la instalación puede continuar sin ninguna intervención.

**archivo de salida.** Archivo de base de datos o de dispositivo que se abre con la opción para permitir la grabación de registros.

**archivo de trabajo.** En el proceso de duplicación de datos de DB2, archivo temporal que el programa Apply utiliza al procesar un conjunto de suscripción.

**archivo de vertido.** En el proceso de duplicación de datos de DB2, archivo temporal creado por el programa Apply que se utiliza como fuente para actualizar datos en varias tablas destino.

**archivo Java.** Formato de archivo que se utiliza para agregar muchos archivos en uno. Se conoce normalmente como archivo *JAR*.

**archivo secuencial.** Archivo no perteneciente a DB2 Universal Database para OS/390 cuyos registros están dispuestos de acuerdo con sus posiciones físicas sucesivas, como por ejemplo las que ocupan en una cinta magnética. Varios de los programas de utilidad para bases de datos de DB2 Universal Database para OS/390 necesitan archivos secuenciales.

**archivo SYS1.DUMPxx.** En un entorno OS/390, archivo que contiene un vuelco del sistema.

**área de comunicaciones compartida (shared communications area, SCA).** Estructura de lista del recurso de acoplamiento que un grupo de compartimiento de datos de DB2 Universal Database para OS/390 utiliza para la comunicación entre sistemas DB2.

**área de comunicaciones de SQL (SQLCA).** Conjunto de variables que proporciona un programa de aplicación con información acerca de la ejecución de las sentencias de SQL o sus peticiones procedentes del gestor de bases de datos.

**área de datos.** Área de memoria que un programa utiliza para contener información.

**área de descriptores de SQL (SQLDA).** (1) Conjunto de variables que se utiliza en el proceso de determinadas sentencias de SQL. La SQLDA se utiliza para programas de SQL dinámico. (2) Estructura que describe las variables de entrada, las variables de salida o las columnas de una tabla resultante.

**área de servicio común (CSA).** En OS/390, parte del área común que contiene las áreas de datos que pueden ser direccionadas por todos los espacios de direcciones.

**área de trabajo de diagnóstico del sistema (system diagnostic work area, SDWA).** En un entorno OS/390, datos registrados en una entrada de SYS1.LOGREC que describen un error de programa o de hardware.



**área temática.** En Centro de depósito de datos, conjunto de procesos que crean datos de depósito para una determinada área lógica de negocio. Los procesos de un área temática operan sobre datos de un determinado tema para crear los datos de detalle, resúmenes de datos y cubos necesarios para ese tema.

**argumento.** Valor que se pasa a una función o procedimiento, o que es devuelto por éstos, durante la ejecución del programa.

**Arquitectura de Bases de Datos Relacionales Distribuidas (DRDA).** Arquitectura que define formatos y protocolos para proporcionar acceso transparente a datos remotos. DRDA define dos tipos de funciones, la función "petionario de aplicaciones" en la página 1592 y la función "servidor de aplicaciones" en la página 1607.

**Arquitectura de red de sistemas (Systems Network Architecture - SNA).** Arquitectura que describe la estructura lógica, los formatos, los protocolos y las secuencias de operación para transmitir unidades de información a través de redes y también secuencias de operación para controlar la configuración y la operación de las redes.

**Arquitectura de representación de datos de tipo carácter (Character Data Representation Architecture - CDRA).** Arquitectura que se utiliza para lograr una representación, proceso e intercambio coherentes de datos de tipo serie.

**arranque en frío.** (1) Proceso de arrancar un sistema o programa utilizando un programa de carga del programa inicial. (2) Proceso mediante el cual DB2 Universal Database para OS/390 rearranca sin procesar ningún registro de archivo de anotaciones. Véase también "arranque en templado".

**arranque en templado.** (1) Reinicio que permite volver a utilizar colas de trabajo de entrada y de salida inicializadas previamente. (2) En el proceso de duplicación de datos de DB2, arranque del programa Capture que permite volver a utilizar colas de trabajo de entrada y de salida inicializadas previamente. Véase también "arranque en frío".

**ASCII.** Esquema de codificación que se utiliza para representar series de caracteres en muchos entornos, normalmente en Personal Computer y estaciones de trabajo. Véase también "EBCDIC" en la página 1556 y "Unicode" en la página 1621.

**asignación de plan.** El proceso de asignar recursos de DB2 Universal Database para OS/390 a un plan para preparar su ejecución.

**asignación de recursos.** En DB2 Universal Database para OS/390, parte del plan de asignación que trata específicamente con los recursos de base de datos.

**asíncrono.** Sin relación temporal regular; imprevisible respecto al proceso de las instrucciones del programa. Véase también "síncrono" en la página 1609.

**asociado de sesión.** En SNA, una de las dos unidades direccionables de red (NAU) que participan en una sesión activa.

**AST.** Véase "tabla de resumen automática" en la página 1615.

**atributo.** En el diseño de bases de datos SQL, característica de una entidad. Por ejemplo, el número de teléfono de un empleado es uno de los atributos del empleado.

## Glosario

**atributo de longitud.** Valor asociado a una serie de caracteres que representa la longitud máxima o la longitud fija declarada de la serie.

**atributo no condensado.** Atributo de tabla que indica que la tabla contiene un histórico de cambios efectuados en los datos, no los datos actuales. Una tabla que tiene establecido este atributo incluye más de una fila para cada valor de clave.

**auditoría.** Registro de información que tiene lugar después de detectar que individuos o aplicaciones acceden a datos supervisados.

**autenticación.** Proceso mediante el cual un sistema verifica la identidad de un usuario. La autenticación de usuario la realiza un recurso de seguridad externo a DB2, que normalmente forma parte del sistema operativo o es un producto independiente.

**autónomo.** Atributo de un programa que significa que es capaz de ejecutarse independientemente de DB2 Universal Database para OS/390, sin utilizar sus servicios.

**autorización.** Véase “nivel de autorización” en la página 1581.

**autorización.** Proceso mediante el cual DB2 obtiene información acerca del usuario autenticado, que indica las operaciones de base de datos que dicho usuario puede realizar y los objetos de datos a los que puede acceder ese usuario. Véase también “privilegio” en la página 1593 y “nivel de autorización” en la página 1581.

**autorización administrativa.** Niveles de autorización SYSADM y DBADM que tienen privilegios completos para los recursos de instancia y los recursos de base de datos respectivamente.

**autorización de carga.** Proporciona al programa de utilidad LOAD o al programa de utilidad AutoLoader privilegios para cargar datos en tablas.

**autorización del sistema.** Niveles de autorización SYSCTRL y SYSMAINT que tienen privilegios completos para gestionar el sistema pero sin la posibilidad de acceder a los datos.

**autorización de uso público.** Autorización para un objeto otorgada a todos los usuarios.

## B

**base de datos de comunicaciones (CDB).** Conjunto de tablas del catálogo de DB2 Universal Database para OS/390 que se utilizan para establecer conversaciones con sistemas remotos de gestión de bases de datos.

**base de datos de control de depósito.** Base de datos del Centro de depósito de datos que contiene las tablas de control necesarias para almacenar metadatos del Centro de depósito de datos.

**base de datos de directorios distribuidos.** Listado completo de todos los recursos de la red que se encuentran en los directorios individuales dispersos en una red APPN. Cada nodo tiene una parte del directorio completo, pero no es necesario que ningún nodo tenga la lista completa. Las entradas se crean, modifican y suprimen mediante definición del sistema, acción del operador, registro automático y los procedimientos continuos de búsqueda en red. Sinónimo de directorio de red distribuido.

**Base de datos de gestor de transacciones (Transaction Manager Database) (Base de datos TM).** Base de datos que se utiliza para registrar transacciones cuando se utiliza una confirmación en dos fases

(SYNCPOINT TWOPHASE) con bases de datos DB2. Si se produce un error de transacción, se puede acceder a la información de la base de datos del Gestor de Transacciones para resincronizar las bases de datos implicadas en la transacción anómala.

**base de datos de modelo.** En el entorno OS/390, base de datos de DB2 que se crea en la estación de trabajo que se utiliza para crear un modelo de un subsistema DB2 en el entorno OS/390, que pueda utilizarse entonces para la optimización de índices y consultas.

**base de datos de registro.** En un entorno OS/390, base de datos de información de seguridad sobre principales, grupos, organizaciones, cuentas y políticas de seguridad.

**base de datos local.** Base de datos que está ubicada físicamente en la estación de trabajo que se está utilizando. Véase también “base de datos remota”.

**base de datos multidimensional.** En el Kit de iniciación de OLAP, base de datos no relacional en la que se pueden copiar datos relacionales para el análisis OLAP. Véase también “cubo relacional” en la página 1547.

**base de datos particionada.** Base de datos con dos o más particiones de base de datos. Los datos de las tablas de usuario pueden ubicarse en una o más particiones de base de datos. Cuando una tabla está en varias particiones, algunas de sus filas se almacenan en una partición y otras se almacenan en otras particiones. Véase “partición de base de datos” en la página 1590.

**base de datos relacional.** Base de datos que se puede percibir como un conjunto de tablas y manipular de acuerdo con el modelo relacional de datos. Cada base de datos incluye un conjunto de tablas de catálogo del sistema que describen la estructura lógica y física de los datos, un archivo de configuración que contiene los valores de parámetros asignados para la base de datos y una anotación cronológica de recuperación con transacciones en curso y transacciones archivables.

**base de datos relacional distribuida.** Base de datos cuyas tablas están guardadas en sistema diferentes pero interconectados.

**base de datos remota.** Base de datos que está ubicada físicamente en una estación de trabajo distinta de la que se está utilizando. Véase también “base de datos local”.

**Base de datos TM.** Véase 1533.

**BLOB.** Véase “gran objeto binario” en la página 1566.

**bloque.** (1) Serie de elementos de datos que se registra o se transmite como una unidad. (2) Conjunto de páginas de datos contiguas de una agrupación de almacenamientos intermedios.

**bloque de consulta.** En DB2 Universal Database para OS/390, parte de una consulta que está representada por una de las cláusulas FROM. Cada cláusula FROM puede tener varios bloques de consulta, dependiendo de la forma en que DB2 Universal Database para OS/390 procesa internamente la consulta.

**bloque de control de tareas (TCB).** Bloque de control que se utiliza para transmitir información sobre tareas de un espacio de direcciones que están conectadas a DB2 Universal Database para OS/390. Un espacio de direcciones puede dar soporte a muchas conexiones de tarea (una por cada tarea), pero solo una conexión de espacio de direcciones.

## Glosario

**bloque de cursor.** Técnica que reduce la actividad general haciendo que el gestor de bases de datos recupere un bloque de filas en una sola operación. Estas filas se almacenan en una antememoria mientras se procesan.

**bloque de datos.** Proceso que consiste en especificar cuántos datos, medidos en minutos de modificación de datos, se duplicarán en un ciclo de suscripción. Véase también “agrupación en bloques” en la página 1527.

**bloqueo.** Mecanismo utilizado por el gestor de bases de datos para asegurar la integridad de los datos. El bloqueo impide que varios usuarios accedan simultáneamente a datos no coherentes.

**bloqueo.** (1) Medio de serializar sucesos o el acceso a datos. (2) Medio de impedir que los cambios no confirmados efectuados por un proceso de aplicación sean percibidos por otro proceso de aplicación y para impedir que un proceso de aplicación actualice datos a los que está accediendo otro proceso.

**bloqueo compartido.** Bloqueo que limita los procesos de aplicación que se ejecutan simultáneamente a operaciones de sólo lectura en los datos de la base de datos. Véase también “bloqueo exclusivo”.

**bloqueo de drenaje.** En DB2 Universal Database para OS/390, bloqueo en una clase de reclamación que impide que se produzca una reclamación.

**bloqueo de fila.** Bloqueo en una fila de datos individual. Véase también “bloqueo” y “bloqueo de tabla”.

**bloqueo de LOB.** En DB2 Universal Database para OS/390, bloqueo en un valor de LOB.

**bloqueo de tabla.** Bloqueo en una tabla de datos. Véase también “bloqueo de fila” y “identificador de fila” en la página 1570.

**bloqueo de tabla global.** Bloqueo de tabla que se obtiene en todas las particiones de un grupo de particiones de base de datos de una tabla.

**bloqueo de tabla local.** Bloqueo de tabla que se establece sólo sobre partición individual de base de datos.

**bloqueo de transacción.** En DB2 Universal Database para OS/390, bloqueo utilizado para controlar la ejecución concurrente de sentencias de SQL.

**bloqueo exclusivo.** Bloqueo que impide que los procesos de aplicación que se ejecutan simultáneamente accedan a datos de la base de datos. Véase también “bloqueo compartido”.

**bloqueo físico (bloqueo P).** Tipo de bloqueo que DB2 Universal Database para OS/390 establece para proporcionar coherencia para datos que están puestos en antememoria en diferentes subsistemas DB2 Universal Database para OS/390. Los bloqueos físicos se utilizan sólo en entornos de compartimiento de datos. Véase también “bloqueo lógico (bloqueo L)” en la página 1535.

**bloqueo general.** En DB2 Universal Database para OS/390, bloqueos de la modalidad *shared*, *update* o *exclusive* sobre una tabla, partición o espacio de tablas.

**bloqueo global.** En DB2 Universal Database para OS/390, bloqueo que proporciona control de simultaneidad dentro y entre subsistemas DB2. El bloqueo abarca a todos los subsistemas DB2 de un grupo de compartimiento de datos.

**bloqueo jerárquico explícito.** En DB2 Universal Database para OS/390, bloqueo utilizado para que IRML conozca la relación padre-hijo existente entre los recursos. Esta clase de bloqueo evita la actividad asociada al bloqueo general cuando no existe ningún interés inter-DB2 para un recurso.

**bloqueo local.** Bloqueo que proporciona control de simultaneidad intra-DB2, pero no control de simultaneidad inter-DB2; su ámbito de actuación es un sistema DB2 Universal Database para OS/390 individual.

**bloqueo lógico (bloqueo L).** En DB2 Universal Database para OS/390, tipo de bloqueo utilizado por las transacciones para controlar la simultaneidad de datos intra-DB2 e inter-DB2 entre transacciones. Véase también “bloqueo físico” en la página 1534.

**bloqueo negociable.** En DB2 Universal Database para OS/390, bloqueo cuya modalidad se puede atenuar, mediante acuerdo entre los usuarios que compiten, para que sea compatible con todos. Un bloqueo físico es un ejemplo de bloqueo negociable.

**bloqueo P.** Véase “bloqueo físico” en la página 1534.

**bloqueo padre.** En el bloqueo jerárquico explícito de DB2 Universal Database para OS/390, bloqueo mantenido sobre un recurso que tiene bloqueos hijo en un nivel inferior de la jerarquía; generalmente los bloqueos de partición o de espacio de tablas son bloqueos padre.

**bloqueo retenido.** Bloqueo de modificación (MODIFY) que un subsistema de DB2 Universal Database para OS/390 estaba manteniendo cuando produjo un error de subsistema. El bloqueo se retiene en la estructura de bloqueo de recurso de acoplamiento cuando se produce un error de DB2 Universal Database para OS/390.

**bloqueos de bloque.** Bloqueo de un bloque en un entorno de cluster de múltiples dimensiones.

**bloqueos de modificación.** En DB2 Universal Database para OS/390, un bloqueo L o bloqueo P con un atributo MODIFY. En la estructura de bloqueo de recurso de acoplamiento se conserva siempre una lista de estos bloqueos activos. Si falla el subsistema que efectúa la petición, los bloqueos de modificación de ese subsistema se convierten en bloqueos retenidos.

**BSAM.** Véase “método de acceso secuencial básico” en la página 1579.

**BSDS.** Véase “conjunto de datos de rutina de carga” en la página 1543.

**búsqueda no sensible a las mayúsculas y minúsculas.** Resultado de una búsqueda en la que no se han tenido en cuenta las mayúsculas y minúsculas de la serie que se está buscando.

## C

**cabecera de anotación cronológica.** Registro de anotaciones cronológicas más antiguo grabado de la anotación cronológica activa.

**cabecera de fila lógica (LRH).** En DB2 Universal Database para OS/390, cada registro lógico incluye un prefijo llamado cabecera de fila lógica, que contiene información de control. Sólo el primer segmento contiene la LRH entera; los segmentos posteriores sólo incluyen los dos primeros campos cuando se necesita una anotación cronológica específica para la recuperación. Todos los segmentos se devuelven y se presentan juntos como si el registro se hubiera almacenado continuamente.

## Glosario

**cadena de seudónimos.** Serie de seudónimos de tabla que hacen referencia entre sí de forma secuencial no repetitiva.

**CAF.** Véase “recurso de conexión de llamada” en la página 1600.

**calificador de Apply.** En el proceso de duplicación de datos de DB2, serie de caracteres que identifica definiciones de suscripción que son exclusivas de cada instancia del programa Apply. Véase también “conjunto de suscripción” en la página 1544.

**campo de definición de intervalo de control (CIDF).** En VSAM, campo situado en los 4 bytes finales de cada intervalo de control; describe el espacio libre, si lo hay, del intervalo de control.

**captación de bloque.** Función de DB2 que recupera (u obtiene) un gran conjunto de filas juntas. Si se utiliza la captación de bloque, se puede reducir de forma significativa el número de mensajes que se envían a través de la red. La captación de bloque sólo se aplica a los cursores que no actualizan datos.

**captación previa.** Leer datos por adelantado y anticipándose a su utilización.

**captación previa de lista.** Método de acceso que hace uso de la captación previa incluso en las consultas que no acceden secuencialmente a los datos. Esto se realiza mediante la exploración del índice y la recopilación de identificadores de registro antes de acceder a las páginas de datos. Entonces se clasifican dichos RID y se realiza una captación previa de los datos utilizando esta lista.

**captación previa por lectura anticipada.** Método para captar previamente páginas mediante la consulta anticipada en una exploración. Esto produce la recuperación asíncrona de páginas aunque dichas páginas no estén ubicadas secuencialmente en el disco. Véase también “precaptación secuencial” en la página 1593 y “captación previa de lista”.

**carácter de desplazamiento a teclado estándar.** Carácter especial de control (X'0F') que se utiliza en sistemas EBCDIC para indicar que los bytes subsiguientes representan caracteres SBCS. Véase también “carácter de desplazamiento a teclado ideográfico”.

**carácter de desplazamiento a teclado ideográfico.** Carácter especial de control especial (X'0E') que se utiliza en sistemas EBCDIC para indicar que los bytes subsiguientes, hasta el siguiente carácter de control de desplazamiento a teclado estándar, representan caracteres DBCS. Véase también “carácter de desplazamiento a teclado estándar”.

**carácter de escape.** Véase “carácter de escape de SQL”.

**carácter de escape de SQL.** Símbolo utilizado para delimitar un identificador delimitado de SQL. El carácter de escape son las comillas dobles, excepto en las aplicaciones COBOL, donde el usuario asigna el símbolo que puede ser un apóstrofo o unas comillas dobles.

**carácter de máscara.** Carácter que se utiliza para representar caracteres opcionales al principio, en medio y al final de un término de búsqueda. Los caracteres de máscara se utilizan normalmente para buscar variaciones de un término en un índice concreto.

**carácter de reconocimiento de mandato (CRC).** Carácter que permite que un operador de consola de MVS o un usuario del subsistema IMS dirija mandatos de DB2 hacia determinados subsistemas de DB2 Universal Database para OS/390.

**carácter de sustitución.** En SQL, carácter exclusivo que durante la conversión de caracteres sustituye a cualquier carácter del programa fuente que no tenga una correspondencia en la representación codificada de destino.

**carácter gráfico.** Un carácter DBCS.

**Característica DB2I para Kanji.** En DB2 Universal Database para OS/390, cinta que contiene los paneles y trabajos que permiten que un sistema visualice paneles de DB2I en Kanji.

**cardinalidad.** Número de filas de una tabla de base de datos.

**cascada.** En Centro de depósito de datos, consiste en ejecutar una secuencia de sucesos. Cuando un paso ("paso" en la página 1590) está conectado en cascada con otro, los pasos se ejecutan de forma secuencial o concurrente. Un paso de proceso pueden también estar conectado en cascada con un programa, el cual se ejecuta cuando el paso de proceso finaliza su ejecución.

**cascada de desencadenantes.** Proceso que se produce cuando la acción desencadenada de un desencadenante hace que se active otro desencadenante.

**CAT.** Véase "tablas disponibles continuamente" en la página 1616.

**catálogo.** Conjunto de tablas y vistas mantenidas por el gestor de bases de datos. Estas tablas y vistas contienen información sobre la base de datos, tal como descripciones de tablas, vistas e índices.

**catálogo de base de datos.** En Centro de depósito de datos, conjunto de tablas que contiene descripciones de objetos de base de datos, tales como tablas, vistas e índices.

**catálogo de información.** Base de datos, gestionada por un Gestor de catálogos de información, que contiene datos descriptivos ("metadatos comerciales" en la página 1579) que ayudan a los usuarios a identificar y localizar los datos y la información que están disponibles para ellos en la organización. Un catálogo de información también contiene "metadatos técnicos" en la página 1579.

**catálogo del sistema.** Véase "catálogo".

**categoría de coste.** Categoría en la que DB2 Universal Database para OS/390 sitúa estimaciones de coste para sentencias de SQL cuando se establecen enlaces lógicos con la sentencia. Una estimación de coste se puede situar en cualquiera de las categorías de coste siguientes:

- A: Indica que DB2 Universal Database para OS/390 tenía información suficiente para efectuar una estimación de coste sin utilizar valores por omisión.
- B: Indica que existe alguna condición que ha obligado a que DB2 Universal Database para OS/390 utilice valores por omisión para su estimación.

La categoría de coste se exterioriza en la columna COST\_CATEGORY de DSN\_STATEMNT\_TABLE cuando se explica una sentencia.

**CCSID.** Véase "identificador de juego de caracteres codificado" en la página 1570.

**CDB.** Véase "base de datos de comunicaciones" en la página 1532.

**CDRA.** Véase "Arquitectura de representación de datos de tipo carácter" en la página 1531.

**CEC.** En un entorno OS/390, complejo electrónico central



## Glosario

**Centro de administración de satélites.** Interfaz de usuario que proporciona soporte administrativo centralizado para satélites.

**Centro de control.** Interfaz gráfica de DB2 que muestra objetos de base de datos (por ejemplos bases de datos y tablas) y las relaciones existentes entre ellos. Desde el Centro de control, el usuario puede efectuar las tareas proporcionadas por las herramientas Programa de utilidad DBA, Visual Explain y Supervisor de rendimiento. Véase también “herramienta DataJoiner Replication Administration” en la página 1567.

**Centro de depósito de datos.** Interfaz gráfica, y el software que hay detrás de la misma, que permite trabajar con los componentes del depósito de datos. Puede utilizar Centro de depósito de datos para definir y gestionar los datos del depósito y los procesos que crean los datos del depósito.

**certificado de Kerberos.** Mecanismo de aplicación transparente que transmite la identidad de un principal iniciador a su destino. Un certificado simple contiene la identidad del principal, una clave de sesión, una indicación de fecha y hora, e información adicional, todo lo cual está protegido por la clave secreta del destino.

**CFRM.** Véase “Coupling Facility Resource Manager” en la página 1546.

**CI.** Véase “intervalo de control” en la página 1574.

**ciclo.** En DB2 Universal Database para OS/390, conjunto de tablas que se pueden ordenar de modo que cada tabla proceda de la anterior y la primera proceda de la última. Por ejemplo, una tabla autorreferente es un ciclo con un solo miembro.

**ciclo de recurrencia.** Ciclo que se produce cuando una selección completa dentro de una expresión de tabla común incluye el nombre de la expresión de tabla común en una cláusula FROM.

**ciclo de referencia.** Conjunto de restricciones de referencia en el que cada tabla del conjunto es descendiente de sí misma.

**ciclo de suscripción.** En el proceso de duplicación de datos de DB2, proceso en el que el programa Apply recupera datos modificados de un conjunto de suscripción determinado, duplica los cambios en la tabla destino y actualiza las correspondientes tablas de control de la duplicación para reflejar el progreso realizado.

**CICS.** Véase “Sistema de Control de la Información del Cliente” en la página 1609.

**CIDE.** Véase “campo de definición de intervalo de control” en la página 1536.

**clase de optimización de consulta.** Conjunto de reglas de reescritura de consulta y técnicas de optimización para compilar consultas.

**clase de reclamación.** En DB2 Universal Database para OS/390, tipo específico de acceso a objeto que puede ser uno de los tipos siguientes: estabilidad del cursor (“estabilidad del cursor” en la página 1560), lectura repetible (“lectura repetible” en la página 1576) o grabación.

**clase de servicio.** En DB2 Universal Database para OS/390, término de VTAM que designa una lista de rutas a través de una red, dispuestas en orden de preferencia para su uso.



**clase de servicio.** En DB2 Universal Database para OS/390, identificador de 8 caracteres que el Gestor de Cargas de Trabajo de MVS utiliza para asociar objetivos de rendimiento del cliente con una hebra de DDF o procedimiento almacenado determinados. También se utiliza una clase de servicio para clasificar trabajos en asistentes de paralelismo.

**cláusula.** En SQL, parte diferenciada de una sentencia, por ejemplo una cláusula SELECT o una cláusula WHERE.

**cláusula CHECK.** En SQL, extensión de las sentencias CREATE TABLE y ALTER TABLE que especifica una restricción de comprobación de tabla.

**clave.** Columna o conjunto ordenado de columnas que se identifica en la descripción de una tabla, un índice o una restricción de referencia. La misma columna puede formar parte de más de una clave.

**clave compuesta.** Conjunto ordenado de columnas de clave de la misma tabla.

**clave de índice.** Grupo de columnas de una tabla utilizado para determinar el orden de las entradas de índice.

**clave de partición.** (1) Conjunto ordenado de una o más columnas de una tabla determinada. Para cada fila de la tabla, se utilizan los valores de las columnas de clave de particionamiento para determinar a qué partición de base de datos pertenece la fila. (2) En el proceso de duplicación, conjunto ordenado de una o más columnas de una tabla determinada. Para cada fila de la tabla fuente, se utilizan los valores de las columnas de clave de particionamiento para determinar a qué tabla destino pertenece la fila.

**clave de unicidad.** Clave que está restringida de forma que no tenga ningún par de valores iguales.

**clave foránea.** Columna o conjunto de columnas que hace referencia a una clave padre. En una base de datos relacional, clave de una tabla que hace referencia a la clave primaria de otra tabla.

**clave padre.** Clave primaria o clave de unicidad que se utiliza en una restricción referencial. Los valores de una clave padre determinan los valores válidos de la clave foránea en la restricción.

**clave primaria.** Clave de unicidad que forma parte de la definición de una tabla. Una clave primaria es la clave padre por omisión de una definición de restricción referencial.

**CLI.** Véase “interfaz de nivel de llamada” en la página 1573.

**cliente.** Cualquier programa (o estación de trabajo en la que se ejecute un programa) que se comunica con un servidor de bases de datos y accede a él. Véase también “peticionario” en la página 1591.

**cliente DB2.** Permite el acceso a una base de datos remota sin conocer su ubicación física. El cliente DB2 determina la ubicación de la base de datos, gestiona la transmisión de peticiones al servidor de bases de datos y devuelve los resultados.

**cliente de base de datos.** Estación de trabajo que se utiliza para acceder a una base de datos situada en un servidor de bases de datos.

**cliente móvil.** Nodo, generalmente un sistema portátil, donde están situados el habilitador móvil, la tabla fuente de duplicación y la tabla destino de duplicación utilizados en un entorno móvil. La modalidad de duplicación móvil se invoca desde el cliente móvil.

## Glosario

**cliette.** Proceso de ejecución larga del componente Live Connection de Net.Data que atiende peticiones procedentes del servidor Web. El Gestor de Conexiones planifica los procesos de cliette para atender estas peticiones.

**CLIST (command list).** Véase “lista de mandatos” en la página 1577.

**CLOB.** Véase “gran objeto de caracteres” en la página 1566.

**CLP.** Véase “Procesador de línea de mandatos” en la página 1594.

**CLPA (create link pack area).** Véase “crear área de módulos residentes” en la página 1547.

**código de intercambio decimal ampliado codificado en binario (EBCDIC).** Véase “EBCDIC” en la página 1556.

**código de país.** Representación de 2 caracteres para el país. Se utiliza para establecer el formato monetario, de fecha y numérico.

**código de razón de terminación anómala.** Código hexadecimal de 4 bytes que identifica, de forma inequívoca, un problema producido en DB2 Universal Database para OS/390.

**código de retorno de SQL.** SQLCODE o SQLSTATE.

**coherencia física.** En DB2 Universal Database para OS/390, estado de una página que no está en un estado de parcialmente cambiada.

**cola del archivo de anotaciones.** Registro de anotaciones escrito más recientemente en un archivo de anotaciones activo.

**cola de tablas.** Mecanismo para transferir filas entre particiones de base de datos. Las colas de tablas son corrientes de filas distribuidas con reglas simplificadas para la inserción y eliminación de filas. Las colas de tablas también se pueden utilizar para entregar filas entre procesos diferentes de una base de datos no particionada.

**colección.** En DB2 Universal Database para OS/390 grupo de paquetes que tienen el mismo calificador.

**colocación de tablas.** Posibilidad de DB2 de reconocer cuándo se accede a los datos para una unión o una subconsulta y si éstos están ubicados en la misma partición del mismo grupo de nodos. Cuando sucede esto, DB2 puede elegir realizar el proceso de unión o subconsulta en la partición donde están almacenados los datos.

**columna de identidad.** Columna que proporciona un modo para que DB2 genere automáticamente un valor numérico para cada fila que se inserta en la tabla. Las columnas de identidad se definen con la cláusula AS IDENTITY. Una tabla no puede tener más de una columna de identidad.

**columna indicadora.** En DB2 Universal Database para OS/390, valor de 4 bytes que se almacena en una tabla base en lugar de una columna LOB.

**columnas correlacionadas.** En SQL, relación entre el valor de una columna y el valor de otra columna.

**combinación.** Operación de SQL que combina los resultados de dos sentencias de selección. Las combinaciones se utilizan normalmente para fusionar listas de valores que se obtienen de varias tablas.

**compartimiento de datos.** Capacidad de dos o más subsistemas de DB2 Universal Database para OS/390 para acceder directamente y modificar un conjunto individual de datos.

**compensación.** En el proceso de una consulta, el servidor federado puede realizar operaciones soportadas por el SQL de DB2 pero no soportadas por el SQL de la fuente de datos.

**compensación de transacción.** Proceso que restaura filas que están afectadas por una transacción confirmada que se rechaza. Cuando se rechaza una transacción confirmada, las filas se restauran al estado en que estaban antes de confirmar la transacción.

**complejo de proceso central (CPC).** En un entorno OS/390, conjunto físico de hardware (por ejemplo un ES/3090) que consta de almacenamiento principal, uno o más procesadores centrales , temporizadores y canales.

**completa.** Atributo de tabla que indica que la tabla contiene una fila para cada valor de clave primaria de interés. Como resultado, se puede utilizar una tabla fuente completa para efectuar una renovación de una tabla destino.

**comprobación de procedencia.** Opción de seguridad de SNA LU 6.2 que define una lista de identificadores de autorización a los que se les permite conectarse a DB2 Universal Database para OS/390 desde una LU asociada.

**comprobación pendiente.** Estado que puede tomar una tabla en el que sólo se permite una actividad limitada en la tabla y no se comprueban restricciones cuando se actualiza la tabla.

**Comunicación Avanzada Programa a Programa (APPC).** (1) Recurso general que caracteriza a la arquitectura LU 6.2 y a sus diversas implantaciones en productos. Véase también “Comunicaciones de interfaz común de programación”. (2) Programa que permite que los programas de transacciones de un sistema se comuniquen con programas de transacciones ubicados en otros sistemas de una red de Arquitectura de redes de sistemas (“Arquitectura de red de sistemas” en la página 1531).

**Comunicación entre procesos (Inter-Process Communication - IPC).** Mecanismo de un sistema operativo que permite a los procesos comunicarse entre sí en el mismo sistema o a través de una red.

**comunicaciones de igual a igual.** Comunicaciones entre dos unidades lógicas SNA (LU) que no están gestionadas por un sistema principal; se utiliza normalmente cuando se hace referencia a los nodos LU 6.2.

**Comunicaciones de interfaz común de programación (Common Programming Interface Communications - CPI-C).** API para aplicaciones que requieren comunicación de programa a programa y que hace uso de LU 6.2 de SNA para crear un conjunto de servicios entre programas. Véase también “Comunicación Avanzada Programa a Programa”.

**con barrera.** Perteneciente a un tipo de función definida por el usuario o procedimiento almacenado que se define para proteger el gestor de bases de datos frente a las modificaciones efectuadas por la función. El gestor de bases de datos se aísla de la función o del procedimiento almacenado mediante una barrera. Véase también “sin barrera” en la página 1609.

**condensada.** Atributo de tabla que indica que la tabla contiene datos actuales en lugar de una historia de los cambios efectuados en los datos. Una tabla condensada incluye una sola fila para cada valor de clave primaria de la tabla. Como resultado, una tabla condensada puede utilizarse para proporcionar información actual para una renovación.

## Glosario

**condición de acción desencadenada.** (1) Condición de búsqueda que controla la ejecución de las sentencias de SQL dentro de la acción desencadenada. (2) Parte opcional de la acción desencadenada. Esta acción booleana aparece como una cláusula WHEN y especifica una condición que DB2 evalúa para determinar si deben ejecutarse las sentencias de SQL desencadenadas.

**condición de búsqueda.** Criterio para seleccionar filas de una tabla. Una condición de búsqueda consta de uno o más predicados.

**condición de control.** Forma restringida de condición de búsqueda que se utiliza en restricciones de control.

**conectada por supresión.** En SQL, una tabla está conectada por supresión con la tabla P si depende de la tabla P o depende de una tabla a la que se propagan en cascada operaciones de supresión desde la tabla P.

**conectar.** En DB2, acceder a objetos a nivel de base de datos.

**conectar.** Acceder de forma remota a objetos a nivel de instancia.

**conexión.** (1) Asociación entre un proceso de aplicación y un servidor de aplicaciones. (2) En comunicación de datos, asociación establecida entre unidades funcionales para transmitir información. (3) En SNA, existencia de una vía de comunicación entre dos LU asociadas que permite intercambiar información (por ejemplo, dos subsistemas DB2 Universal Database para OS/390 que están conectados y se comunican mediante una conversación).

**conexión de instancia DBMS.** Conexión lógica entre una aplicación y un proceso agente o hebra que es propiedad de una instancia de DB2.

**conexión dirigida por el sistema.** Conexión que un RDBMS gestiona mediante el proceso de sentencias de SQL con nombres de tres partes (o apodos), proporcionando un nivel de transparencia de ubicación. Véase también “conexiones dirigidas por aplicación”.

**conexiones dirigidas por aplicación.** Las aplicaciones gestionan las conexiones utilizando la sentencia CONNECT de SQL. Véase también “conexión dirigida por el sistema”.

**conexión por protocolo privado.** Conexión privada a DB2 del proceso de aplicación. Por ejemplo, si la primera fase del programa de aplicación utiliza el acceso a protocolo privado DB2 y la segunda fase utiliza el acceso DRDA, las conexiones por protocolo privado DB2 abiertas de la primera fase pueden hacer que falle una operación CONNECT en la segunda fase. Véase también “conexión privada”.

**conexión privada.** Conexión de comunicaciones que es específica de DB2 Universal Database para OS/390. Por ejemplo, cuando el servidor de aplicaciones es un subsistema DB2, se asignan las conexiones DB2 privadas que son necesarias para dar soporte a las referencias a objetos en otros subsistemas DB2. Al igual que las conexiones SQL, las conexiones DB2 privadas están inicialmente en estado retenido y pueden colocarse en estado pendiente de liberación.

**conexión SQL.** Asociación entre un proceso de aplicación y un servidor de aplicaciones local o remoto.

**confirmación.** Operación que finaliza una unidad de trabajo mediante la liberación de bloqueos para que los cambios efectuados por esa unidad de trabajo en la base de datos puedan ser percibidos por otros procesos. Esta operación hace que los cambios efectuados en los datos sean permanentes.

**confirmación automática.** Confirmar automáticamente la unidad de trabajo actual después de cada sentencia de SQL.

**confirmación de petición.** En DB2 Universal Database para OS/390, voto que se somete en la fase de preparación si el participante ha modificado datos y está preparado para confirmar o retrotraer.

**confirmación en dos fases.** Proceso de dos pasos mediante el cual se confirman los recursos recuperables y un subsistema externo. Durante el primer paso, se sondean los subsistemas del gestor de bases de datos para asegurarse de que están listos para confirmarse. Si todos los subsistemas responden positivamente, el gestor de bases de datos les indica que efectúen la confirmación. Véase también “transacción distribuida” en la página 1620.

**conjunto de datos de rutina de carga (BSDS).** Conjunto de datos VSAM que contiene información de nombres y estado para DB2 Universal Database para OS/390, así como especificaciones de rango de direcciones de byte relativo para todos los conjuntos de datos de anotación cronológica activos y de archivador. También contiene las contraseñas para el directorio y catálogo de DB2 Universal Database para OS/390 y listas de registros de re arranque condicional y de punto de control.

**conjunto de datos lineal (LDS).** En un entorno OS/390, conjunto de datos VSAM que contiene datos, pero ninguna información de control. Se puede acceder a un conjunto de datos lineal como una serie direccionable por byte situada en almacenamiento virtual.

**conjunto de datos ordenado por clave (KSDS).** En un entorno OS/390, archivo o conjunto de datos de VSAM cuyos registros se cargan en secuencia de clave y está controlado por un índice.

**conjunto de datos particionado (PDS).** En un entorno OS/390, archivo de datos situado en almacenamiento de acceso directo que está dividido en particiones llamadas miembros. Cada partición puede contener un programa, parte de un programa o datos. Sinónimo de *biblioteca de programas*.

**conjunto de espacios de tablas.** En DB2 Universal Database para OS/390, conjunto de espacios de tablas y particiones que deben recuperarse juntos si cada uno contiene una tabla que es padre o descendiente de una tabla de otro de ellos o el conjunto contiene una tabla base y tablas auxiliares asociadas. Un conjunto de espacios de tablas puede contener ambos tipos de relaciones.

**conjunto de páginas.** En un entorno OS/390, forma alternativa de designar un espacio de tablas o espacio de índice. Cada conjunto de páginas consta de una colección de archivos VSAM.

**conjunto de páginas particionado.** En un entorno OS/390, un espacio de tablas particionado o un espacio de índice particionado. Las páginas de cabecera, las páginas de correlación de espacio, las páginas de datos y las páginas de índice hacen referencia sólo a los datos situados dentro del ámbito de la partición.

**conjunto de páginas simple.** En DB2 Universal Database para OS/390, un conjunto de páginas no particionado. Un conjunto de páginas simple inicialmente consta de un único conjunto de datos (parte del conjunto de páginas). Si dicho conjunto de datos se amplía a 2 gigabytes menos 1 byte, se creará otro conjunto de datos y así sucesivamente hasta un total de 32 conjuntos de datos. DB2 Universal Database para OS/390 considera los conjuntos de datos como un solo espacio de direcciones lineal contiguo que contiene un máximo de 64 gigabytes. Los datos se almacenan en la siguiente posición disponible de este espacio de direcciones, sin tener en cuenta ningún esquema de partición.

## Glosario

**conjunto de privilegios.** En el ID SYSADM de instalación, conjunto de todos los privilegios posibles. Para cualquier otro identificador de autorización, conjunto de todos los privilegios que están registrados para dicho identificador en el catálogo de DB2 Universal Database para OS/390.

**conjunto de restauración.** Copia de seguridad de una base de datos o un espacio de tablas más cero o más archivos de anotaciones cronológicas que, cuando se restaura y se recupera en avance, vuelve a dejar la base de datos o el espacio de tablas en un estado coherente.

**conjunto de suscripción.** En el proceso de duplicación de datos de DB2, especificación de un grupo de tablas fuente, tablas destino y de la información de control que dirige la duplicación de datos cambiados. Véase también “miembro de conjunto de suscripción” en la página 1580 y “fuente de duplicación” en la página 1563.

**conjunto resultante.** Conjunto de filas devueltas por un procedimiento almacenado.

**conmutador de supervisor.** Parámetros del gestor de la base de datos manipulados por el usuario para controlar el tipo de información y la cantidad de información que devuelven las instantáneas de rendimiento.

**constante.** Elemento de lenguaje que especifica un valor que no cambia. Las constantes pueden ser constantes de tipo serie o constantes numéricas. Véase también “variable” en la página 1624.

**consulta.** (1) Petición de información a la base de datos de acuerdo con unas condiciones específicas; por ejemplo, una petición de una lista de todos los clientes de una tabla de clientes cuyo saldo sea mayor que 150.000 Pts. (2) En DB2 Universal Database para OS/390, componente de ciertas sentencias de SQL que especifica una tabla resultante.

**consulta recurrente.** Selección completa que utiliza una expresión de tabla común recurrente.

**contador.** Representación de información que es acumulativa hasta que se toma la muestra. Cuenta valores que aumentan, por ejemplo el número de puntos muertos. Los contadores se restablecen a cero cuando se detiene y se reinicia una instancia de base de datos o una base de datos. Véase también “marcador” en la página 1578.

**contención.** En el gestor de bases de datos, situación en la que una transacción intenta bloquear una fila o tabla que ya está bloqueada.

**contención de bloqueo físico.** En DB2 Universal Database para OS/390, estados conflictivos de los peticionarios de un bloqueo físico. Véase también “bloqueo negociable” en la página 1535.

**contención de bloqueo global.** En DB2 Universal Database para OS/390 conflictos en peticiones de bloqueo entre diferentes miembros de un grupo de compartimiento de datos cuando esos miembros intentan serializar recursos compartidos.

**contención falsa por bloqueo global.** En DB2 Universal Database para OS/390, indicación de contención procedente del recurso de acoplamiento que se produce cuando varios nombres de bloqueo se calculan aleatoriamente al mismo indicador y no existe ninguna contención real.

**contenedor.** (1) Ubicación de almacenamiento físico de los datos. Por ejemplo, un archivo, un directorio o un dispositivo. (2) Véase 1544.

**contenedor de espacio de tablas.** Asignación de espacio a un espacio de tablas. Según el tipo de espacio de tablas, el contenedor puede ser un directorio, un dispositivo o un archivo.

**controlador de ODBC.** Controlador que implementa llamadas de función ODBC e interactúa con una fuente de datos.

**control de confirmación.** Establecimiento de un límite de dentro del proceso bajo el que se está ejecutando Net.Data, donde las operaciones sobre recursos son parte de una unidad de trabajo.

**control de enlace de datos (DLC).** En SNA, capa de protocolo que consta de las estaciones de enlace que planifican la transferencia de datos en un enlace entre dos nodos y efectúan el control de errores para el enlace.

**conversación.** En APPC, conexión entre dos programas de transacciones mediante una sesión entre unidades lógicas (LU-a-LU) que les permite comunicarse entre sí mientras procesan una transacción.

**conversación básica.** Conversación SNA LU 6.2 entre dos programas de transacciones que utilizan la API de conversación básica de APPC. Véase también “conversación correlacionada”.

**conversación correlacionada.** En APPC, conversación entre dos programas de transacciones (TP) que utilizan la API de conversación correlacionada de APPC. En las situaciones normales, los TP de usuario final utilizan conversaciones correlacionadas y los TP de servicio utilizan conversaciones básicas. Ambos tipos de programa pueden utilizar cualquiera de los dos tipos de conversación. Véase también “conversación básica”.

**conversación de proceso SQL.** Cualquier conversación que necesite acceder a datos de DB2 Universal Database para OS/390 mediante una aplicación o peticiones de consulta dinámica.

**conversación de sistemas.** Conversación que deben establecer dos subsistemas de DB2 Universal Database para OS/390 para procesar mensajes del sistema a fin de comenzar un proceso de datos distribuido.

**conversación protegida.** En un entorno OS/390, conversación VTAM que da soporte a los flujos de confirmación en dos fases.

**conversión de caracteres.** Proceso mediante el cual se cambian datos de una representación de codificación de caracteres a otra.

**conversión de contracción.** Proceso que se produce cuando la longitud de una serie convertida es menor que la de la serie fuente. Véase también “conversión de expansión”.

**conversión de expansión.** Se produce cuando la longitud de la serie convertida es mayor que la de la serie fuente. Véase también “conversión de contracción”.

**coordinador.** En DB2 Universal Database para OS/390, componente del sistema que coordina la confirmación o la retrotracción de una unidad de trabajo que incluye trabajo que se realiza en uno o más sistemas diferentes.

**copia continua.** Técnica de recuperación en la que no se escribe nunca encima del contenido de la página actual. En lugar de ello, se asignan y se escriben páginas nuevas mientras que las páginas cuyos valores se están sustituyendo se conservan como copias duplicadas hasta que ya no sean necesarias para permitir la restauración del estado del sistema debido a una retrotracción de la transacción.

**copia de carga.** Imagen de copia de seguridad de datos que se han cargado anteriormente y que se pueden restaurar durante la recuperación ascendente.



## Glosario

**copia de seguridad delta.** Copia de todos los datos de base de datos que han cambiado desde que se ha realizado la última copia de seguridad satisfactoria (completa, incremental o delta) del espacio de tablas en cuestión. También se conoce como imagen de copia de seguridad no acumulativa o diferencial. El predecesor de una imagen de copia de seguridad delta es la copia de seguridad satisfactoria más reciente que contiene una copia de cada uno de los espacios de tablas de la imagen de copia de seguridad delta.

**copia de seguridad en línea.** Copia de seguridad de la base de datos o del espacio de tablas que se efectúa mientras otras aplicaciones están accediendo a la base de datos o al espacio de tablas. Véase también “copia de seguridad fuera de línea”.

**copia de seguridad fuera de línea.** Copia de seguridad de la base de datos o del espacio de tablas efectuada cuando las aplicaciones no estaban accediendo a la base de datos o al espacio de tablas. Durante una copia de seguridad fuera de línea, el programa de utilidad de copia de seguridad de base de datos obtiene el uso exclusivo de la base de datos hasta que la copia de seguridad se ha completado. Véase también “copia de seguridad en línea”.

**copia de seguridad incremental.** Copia de todos los datos de la base de datos que se han modificado desde la operación de copia de seguridad completa satisfactoria más reciente. También se conoce como imagen de copia de seguridad acumulativa, porque cada una de las copias de seguridad incrementales de una serie realizadas tendrá, con el tiempo, el contenido de la imagen de copia de seguridad incremental anterior. El predecesor de una imagen de copia de seguridad incremental es siempre la copia de seguridad completa satisfactoria más reciente del mismo objeto.

**copia de seguridad pendiente.** Estado de una base de datos o espacio de tablas que impide efectuar una operación hasta que se realiza una copia de seguridad de la base de datos o del espacio de tablas.

**copia imagen.** Reproducción exacta de todo o parte de un espacio de tablas. DB2 Universal Database para OS/390 proporciona programas de utilidad para realizar copias imagen completas (copia del espacio de tablas completo) o copias imagen incrementales (copia de sólo las páginas que se han modificado desde la última copia imagen).

**coprocesador de sentencias de SQL.** En el entorno OS/390, alternativa al precompilador de DB2 que permite al usuario procesar sentencias de SQL en el tiempo de compilación. El usuario invoca un coprocesador de sentencias de SQL especificando una opción de compilador.

**correlación de funciones.** Asociación definida entre dos funciones compatibles, una soportada por la base de datos federada y otra soportada por una fuente de datos.

**correlación de usuario.** Asociación entre la autorización con la que un usuario se conecta a un servidor federado y la autorización con la que el usuario se conecta a una fuente de datos.

**coste.** Uso de recursos totales estimado necesario para ejecutar el plan de acceso para una sentencia (o los elementos de una sentencia). El coste se obtiene de una combinación de coste de CPU (en número de instrucciones) y de E/S (en números de búsquedas y transferencias de páginas).

**Coupling Facility Resource Manager.** En un entorno OS/390, gestiona todos los recursos de acoplamiento en un Sysplex.

**CP.** Véase “punto de control” en la página 1597.

**CPC.** Véase “complejo de proceso central” en la página 1541.



**CPI-C.** Véase “Comunicaciones de interfaz común de programación” en la página 1541.

**CRC.** Véase “carácter de reconocimiento de mandato” en la página 1536.

**CRCR.** En DB2 Universal Database para OS/390, registro de control para el rearranque condicional. Véase “rearranque condicional” en la página 1598.

**crear área de módulos residentes (CLPA).** Opción utilizada durante la carga del programa inicial para inicializar el área paginable de módulos residentes.

**CS.** Véase “estabilidad del cursor” en la página 1560.

**CSA.** Véase “área de servicio común” en la página 1530

**cubo relacional.** Conjunto de datos y metadatos que juntos definen una base de datos multidimensional. Un cubo relacional es la porción de una base de datos multidimensional que se almacena en una base de datos relacional. Véase también “base de datos multidimensional” en la página 1533.

**cuenta de reclamaciones.** En DB2 Universal Database para OS/390, cuenta del número de agentes que están accediendo a un objeto.

**cuerpo de función.** Parte de código que implementa una función.

**cuerpo del desencadenante.** Conjunto de sentencias de SQL que se ejecuta cuando se activa un desencadenante y la condición de acción desencadenada se evalúa como verdadera.

**cursor.** Estructura de control definida que un programa de aplicación utiliza para apuntar a una fila determinada dentro de un conjunto ordenado de filas. El cursor se utiliza para recuperar filas de un conjunto.

**cursor ambiguo.** (1) Un cursor es ambiguo si se cumplen todas las condiciones siguientes:

- La sentencia-select se prepara de forma dinámica
- La sentencia-select no incluye la cláusula FOR READ ONLY ni la cláusula FOR UPDATE
- La opción de enlace LANGLEVEL es SAA1
- Por lo demás, el cursor satisface las condiciones de un cursor suprimible

Un cursor ambiguo se considera de sólo lectura si la opción de enlace BLOCKING es ALL; de lo contrario, se considera suprimible. (2) En DB2 Universal Database para OS/390, cursor de base de datos que no está definido por las cláusulas FOR FETCH ONLY ni FOR UPDATE OF, no está definido en una tabla resultante de sólo lectura, no es el destino de una cláusula WHERE CURRENT en una sentencia UPDATE o DELETE de SQL y que está en un plan o paquete que contiene sentencias PREPARE o EXECUTE IMMEDIATE de SQL. Véase también “cursor no ambiguo” en la página 1548.

**cursor asignado.** Cursor que se define para conjuntos de resultados de procedimientos almacenados utilizando la sentencia ALLOCATE CURSOR de SQL.

**cursor desplazable.** Cursor que se puede mover hacia adelante y hacia atrás. Véase también “cursor no desplazable” en la página 1548.

**cursor insensible.** Especifica que el cursor no tiene sensibilidad para las inserciones, actualizaciones o supresiones realizadas en la base de datos (filas subyacentes de la tabla resultante) una vez que se

## Glosario

materializa la tabla resultante (una fila de tabla resultante se materializa cuando se capturan de la base de datos los valores de la fila). En consecuencia, el tamaño de la tabla resultante, el orden de las filas y los valores para cada fila no cambian después de abrir el cursor. La sentencia SELECT no puede contener una cláusula FOR UPDATE y el cursor no se puede utilizar para actualizaciones o supresiones posicionadas. Una operación UPDATE o DELETE posicionada utilizando una cursor INSENSITIVE desplazable produce un error. Véase también “cursor sensible”.

**cursor no ambiguo.** Cursor que permite a una base de datos relacional determinar si se puede utilizar el bloqueo con el conjunto resultante. Un cursor definido como FOR FETCH ONLY o FOR READ ONLY se puede utilizar con el bloqueo, mientras que un cursor definido como FOR UPDATE no se puede utilizar con el bloqueo. Véase también “cursor ambiguo” en la página 1547.

**cursor no desplazable.** Cursor que sólo se puede mover hacia adelante. Los cursores no desplazables se denominan a veces cursores sólo hacia adelante o cursores serie. Véase también “cursor desplazable” en la página 1547.

**cursor sensible.** Tipo de cursor que es sensible a los cambios efectuados en la base de datos después de que se haya materializado la tabla resultante. Véase también “cursor insensible” en la página 1547.

**cursor sensible STATIC.** El orden de las filas y el tamaño de la tabla resultante son estáticos. El tamaño de la tabla resultante no aumenta después de que se hayan materializado las filas. El orden de las filas se establece cuando se materializa la tabla resultante. Las filas recién insertadas no están visibles para los cursores SENSITIVE STATIC una vez que las filas de la tabla resultante se han materializado. Las filas de la tabla resultante no se mueven si las columnas de la cláusula ORDER BY se actualizan en filas que ya se han materializado.

Los cursores estáticos tienen visibilidad para los cambios efectuados por el cursor utilizando UPDATE WHERE CURRENT OF o DELETE WHERE CURRENT OF. La visibilidad de los cambios efectuados fuera del cursor depende del tipo de FETCH que se utiliza con un cursor SENSITIVE STATIC.

**cursor serie.** Cursor que sólo se puede mover hacia adelante.

## D

**DAD.** Véase “Definición de acceso a documento” en la página 1550.

**DARI.** Véase “Interfaz remota de aplicaciones de bases de datos” en la página 1574.

**Data Facility Storage Management Subsystem (DFSMS).** Componente funcional de DFSMS/MVS o elemento base de OS/390, utilizado para la copia de seguridad y la recuperación de datos así como para la gestión de espacio en volúmenes en la jerarquía de almacenamiento.

**DataJoiner.** Producto, adquirible por separado, que proporciona acceso integrado a datos distribuidos para aplicaciones clientes así como una imagen única de la base de datos en un entorno heterogéneo. Mediante DataJoiner, una aplicación cliente puede unir datos (mediante una sola sentencia de SQL) que están distribuidos entre varios sistemas de gestión de bases de datos o actualizar una sola fuente de datos remota como si los datos fueran locales.

**DATALINK.** Tipo de datos de DB2 que permite que las referencias lógicas de la base de datos a un archivo se guarden fuera de la base de datos.

**Data Links File Filter (DLFF).** Filtro que refuerza la integridad de los datos asegurando el acceso válido a un archivo bajo el control de Data Links Manager.

**Data Links File Manager (DLFM).** Aplicación de SQL que funciona junto con DB2 para gestionar archivos externos a una base de datos.

**datos actuales.** En DB2 Universal Database para OS/390, datos de una estructura de lenguaje principal que son actuales (idénticos) respecto a los datos de la tabla base.

**datos de bit.** Datos de tipo carácter CHAR o VARCHAR no asociados a un juego de caracteres codificado y que por tanto no se convierten nunca.

**DBA.** Véase “administrador de bases de datos” en la página 1526.

**DB2 Application Development Client (DB2 SDK).** Conjunto de herramientas que ayudan a los desarrolladores a crear aplicaciones de base de datos.

**DBCLOB.** Véase “gran objeto de caracteres de doble byte” en la página 1566.

**DB2 Connect.** Producto que permite a las aplicaciones cliente leer y actualizar datos que están almacenados en servidores de aplicaciones DRDA.

**DBCS.** Véase “juego de caracteres de doble byte” en la página 1575.

**DBD.** Véase “descriptor de base de datos” en la página 1551.

**DB de IMS.** Base de datos del sistema de gestión de información.

**DB2I.** En DB2 Universal Database para OS/390, DATABASE 2 Interactive.

**DBID (database identifier).** En DB2 Universal Database para OS/390, identificador de base de datos.

**DB2 Life Sciences Data Connect.** Sistema de middleware de bases de datos. DB2 Life Sciences Data Connect permite ejecutar una sola consulta en una base de datos virtual, cuyos datos subyacentes pueden almacenarse en múltiples fuentes de datos de la industria de ciencias de la vida.

**DBMS (database management system).** Véase “sistema de gestión de bases de datos” en la página 1610.

**DB2 Net Search Extender.** Proporciona la recuperación de texto completo mediante un procedimiento almacenado de DB2. Se ha optimizado principalmente para el rendimiento. La realización de búsquedas utilizando DB2 Net Search Extender puede ser especialmente ventajosa en aplicaciones donde el rendimiento de la búsqueda en índices grandes y la escalabilidad de acuerdo con las consultas concurrentes son factores importantes.

**DB2 PM.** En DB2 Universal Database para OS/390, DATABASE 2 Performance Monitor (supervisor de rendimiento de DATABASE 2).

**DB2 Relational Connect.** Producto utilizado en un sistema federado para consultar y recuperar datos ubicados en otros DBMS, por ejemplo Oracle, Sybase, Microsoft SQL Server, y miembros de DB2 Universal Database, por ejemplo DB2 para OS/390, DB2 para OS/400 y DB2 para Windows.

**DBRM.** Véase “módulo de petición de base de datos” en la página 1581.

## Glosario

**DB2 SDK.** Véase “Cliente de desarrollo de aplicaciones de DB2” en la página 1549.

**DB2 Spatial Extender.** Programa utilizado para crear un sistema de información geográfica (GIS).

**DB2 Text Extender.** Sistema de recuperación de texto completo integrado en DB2 Universal Database. Proporciona potentes características de búsqueda mejoradas con una rica funcionalidad lingüística adicional para aplicaciones con documentos altamente estructurados donde la necesidad de información es compleja y la calidad y la precisión del resultado de la búsqueda son temas clave que están por encima de los tiempos de respuesta del sistema.

**DB2 XML Extender.** Programa utilizado para almacenar y gestionar documentos XML en tablas DB2. Se pueden generar documentos XML bien formados y validados a partir de datos relacionales existentes, almacenar dichos documentos como datos de columna y el contenido de atributos y los elementos XML puede almacenarse en tablas de DB2.

**DCLGEN.** Véase “generador de declaraciones” en la página 1565.

**DDF.** Véase “recurso de datos distribuidos” en la página 1600.

**DDL.** Véase “lenguaje de definición de datos” en la página 1576.

**ddname.** Véase “nombre de definición de datos (ddname)” en la página 1583.

**debe finalizar.** Estado durante el proceso de DB2 Universal Database para OS/390 en el cual debe finalizar la operación completa para mantener la integridad de los datos.

**Definición de acceso a documento (Document Access Definition - DAD).** Definición que se utiliza para habilitar una columna de XML Extender de una colección XML, que está formateada por XML.

**definición de servicio.** En un sistema de bases de datos federado, el DBA proporciona al servidor federado una descripción de cada fuente de datos.

**definición en línea de recursos.** En un entorno OS/390 con CICS, característica que el usuario utiliza para definir en línea recursos de CICS sin ensamblar tablas.

**definidor de función.** En DB2 Universal Database para OS/390, identificador de autorización del propietario del esquema de la función que se especifica en la sentencia CREATE FUNCTION.

**delimitador.** Carácter o distintivo que agrupa o delimita elementos de datos.

**delimitador de serie del SQL.** En DB2 Universal Database para OS/390, símbolo utilizado para delimitar una constante de tipo de serie del SQL. El delimitador de serie del SQL es el apóstrofo ('), excepto en las aplicaciones COBOL, donde el usuario asigna el símbolo que puede ser un apóstrofo o unas comillas dobles (").

**delimitador de series de caracteres.** Caracteres que se utilizan para delimitar series de caracteres en archivos ASCII delimitados que se importan o exportan. Véase también “delimitador”.

**dependiente.** En SQL, referente a un objeto (fila, tabla o espacio de tablas) que tiene como mínimo un padre. Véase también “fila padre” en la página 1563, “tabla padre” en la página 1616 y “espacio de tablas padre” en la página 1559.

**dependiente de GBP.** En DB2 Universal Database para OS/390, estado de un conjunto de páginas o de una partición de conjunto de páginas que depende de la agrupación de almacenamientos intermedios de grupo. Existe un interés activo de lectura/escritura entre subsistemas DB2 para este conjunto de páginas o bien el conjunto de páginas ha cambiado páginas de la agrupación de almacenamientos intermedios de grupo que todavía no se han convertido a DASD.

**depósito.** Véase “depósito de datos”.

**depósito de datos.** (1) Colección de datos no volátil orientada hacia un tema que se utiliza para soportar la toma de decisiones estratégicas. El depósito es el punto central de la integración de datos para la información de una empresa. Es la fuente de datos de las despensas de datos de una empresa y proporciona una vista común de los datos de la empresa. (2) Depósito central de todos los datos o de partes significativas de los mismos que reúne el sistema comercial de una organización. También conocido como *depósito de información*. Véase también “despensa de datos” en la página 1553.

**depurar.** Proceso de manipular los datos extraídos de sistemas operativos para hacerlos utilizables por el depósito de datos.

**desagrupación parcial.** Una tabla puede particionarse en un subconjunto de particiones del sistema. Las tablas no se tienen que particionar en todas las particiones del sistema.

**desbloquear.** Acto de liberar un objeto o recurso del sistema que se había bloqueado previamente y hacer que esté disponible de forma general dentro de DB2 Universal Database para OS/390.

**descendiente.** Relativo a un objeto que depende de un objeto o que depende de un descendiente de un objeto.

**descriptor.** (1) Variable que representa una estructura interna dentro de un sistema de software. (2) Serie de caracteres, creada por un expansor, que se utiliza para representar un objeto de imagen, de audio o de vídeo en una tabla. El descriptor correspondiente a un objeto se guarda en una tabla de usuario y en tablas de soporte administrativo. De esta forma, un expansor puede enlazar el descriptor contenido en una tabla de usuario con información sobre el objeto almacenada en las tablas de soporte administrativo. (3) Valor binario que identifica un documento de texto. Se crea un descriptor para cada documento de texto de una columna de texto cuando Text Extender *habilita* dicha columna para que se utilice.

**descriptor de base de datos (DBD).** Representación interna de una definición de base de datos de DB2 Universal Database para OS/390, que refleja la definición de datos contenida en el catálogo de DB2 Universal Database para OS/390. Los objetos definidos en un descriptor de base de datos son espacios de tablas, tablas, índices, espacios de índice y relaciones.

**descriptor de contexto de conexión.** En la CLI de DB2, objeto de datos que contiene información asociada con una conexión a una fuente de datos. Esta información incluye información general de estado, el estado de la transacción e información de diagnóstico. Véase también “descriptor de contexto de sentencia”.

**descriptor de contexto de sentencia.** En CLI, descriptor de contexto que hace referencia al objeto de datos que contiene información acerca de una sentencia de SQL. Esto incluye información como, por ejemplo, argumentos dinámicos, enlaces lógicos para argumentos dinámicos y columnas, información de cursor, valores de resultados e información de estado. Cada descriptor de contexto de sentencia está asociado con un “descriptor de contexto de conexión”.

## Glosario

**descriptor de entorno.** Descriptor que identifica el contexto global para el acceso a la base de datos. Todos los datos aplicables a todos los objetos del entorno se asocian mediante este descriptor.

**desencadenante.** (1) Objeto de una base de datos que invoca indirectamente el gestor de bases de datos cuando se ejecuta una sentencia de SQL determinada. (2) Conjunto de sentencias de SQL que se almacena en una base de datos DB2 y se ejecuta cuando se produce un suceso determinado en una tabla de DB2 .

**desencadenante anterior.** Desencadenante que está definido con la hora de activación de desencadenante BEFORE. Véase también “desencadenante”.

**desencadenante de actualización.** En DB2 Universal Database para OS/390, desencadenante que está definido mediante la operación desencadenante UPDATE de SQL.

**desencadenante de Capture.** En duplicación de DB2, mecanismo que captura las operaciones de supresión, actualización e inserción realizadas en tablas fuente no DB2. Véase también “programa Capture” en la página 1595 y “programa Apply” en la página 1595.

**desencadenante de fila.** En DB2 Universal Database para OS/390, desencadenante cuya granularidad está definida como FOR EACH ROW.

**desencadenante de inserción.** Desencadenante que se define con la operación de SQL desencadenante INSERT.

**desencadenante de sentencia.** En DB2 Universal Database para OS/390, desencadenante cuya granularidad está definida como FOR EACH STATEMENT. Véase también “desencadenante”.

**desencadenante de supresión.** Desencadenante que se define con la operación de SQL desencadenante DELETE. Véase “desencadenante”.

**desencadenante de umbral.** Suceso que se produce cuando el valor de una variable de rendimiento excede o está por debajo de un valor umbral definido por el usuario. La acción que se produce como resultado de un desencadenante de umbral puede ser:

- Anotar información en un archivo de anotaciones de alertas.
- Visualizar información en una ventana de anotaciones de alertas.
- Generar una alarma sonora.
- Abrir una ventana de mensajes.
- Invocar un programa o mandato predefinido.

**desencadenante no operativo.** Desencadenante que depende de un objeto que se ha eliminado o devenido no operativo o de un privilegio que se ha revocado. Véase “desencadenante”.

**deshacer.** (1) Recuperar la última edición que ha tenido lugar. (2) Estado de una unidad de recuperación que indica que deben cancelarse los cambios realizados por la unidad de recuperación en recursos recuperables de DB2 Universal Database para OS/390.

**desnormalización.** Paso clave para designar un diseño de base de datos relacional física. La desnormalización es la duplicación intencionada de columnas en múltiples tablas y la consecuencia es el aumento de redundancia de datos. La desnormalización es necesaria a veces para minimizar problemas de rendimiento. Véase también “normalización” en la página 1585.

**despensa de datos.** Subconjunto de un depósito de datos que contiene datos adaptados a las necesidades específicas de un departamento o equipo de trabajo. Una despensa de datos puede ser un subconjunto de un depósito de la organización entera, por ejemplo los datos contenidos en las herramientas OLAP.

**desplazamiento.** En el entorno OS/390, posibilidad de utilizar un cursor para ir a buscar hacia adelante o hacia atrás. La sentencia FETCH soporta múltiples orientaciones de búsqueda para indicar la nueva posición del cursor. Véase también “orientación de fetch” en la página 1587.

**destino.** En Centro de depósito de datos, tabla, vista o archivo que es creado o llenado con datos por un paso de proceso; constituye la salida de un paso de proceso. Véase también “fuente” en la página 1563.

**destino de depósito.** Subconjunto de tablas, índices y seudónimos de una base de datos individual que son gestionados por el Centro de depósito de datos.

**detección de conflicto.** (1) En configuraciones de duplicación de actualización en cualquier lugar (update-anywhere), proceso de detección de errores de restricción. (2) Proceso de detectar si la misma fila se actualizó en las tablas fuente y destino durante el mismo ciclo de duplicación. Cuando se detecta un conflicto, se rechaza la transacción que lo ha causado. Véase también “detección mejorada de conflictos”, “detección estándar de conflictos” y “detección de conflictos de duplicación de fila”.

**detección de conflictos de duplicación de fila.** En el proceso de duplicación de datos de DB2, detección de conflictos realizada para cada fila, no para cada transacción, cómo ocurre en los duplicados de DB2. Véase también “detección de conflicto”, “detección mejorada de conflictos” y “detección estándar de conflictos”.

**detección estándar de conflictos.** Detección de conflictos en la que el programa Apply busca conflictos en filas que ya están capturadas en las tablas de datos de cambios de la tabla de duplicado o tabla de usuario. Véase también “detección de conflicto”, “detección mejorada de conflictos” y “detección de conflictos de duplicación de fila”.

**detección mejorada de conflictos.** Detección de conflictos que asegura la integridad de los datos entre todos los duplicados y la tabla fuente. El programa Apply bloquea todos los duplicados o tablas de usuario del conjunto de suscripción para impedir transacciones ulteriores. El programa Apply comienza la detección después de que se hayan capturado todos los cambios efectuados antes del bloqueo. Véase “detección de conflicto”.

**detector de puntos muertos.** Proceso dentro del gestor de bases de datos que supervisa los estados de los bloqueos para determinar si existe una condición de punto muerto. Cuando se detecta una condición de punto muerto, el detector detiene una de las transacciones implicadas en el punto muerto. Esta transacción se retrotrae y la otra transacción puede continuar.

**DFF.** En un entorno OS/390, Data Facility Product.

**diccionario.** Colección de información lingüística relacionada con el idioma que Text Extender utiliza durante el análisis de textos, la indexación, la recuperación y el resaltado de documentos en un idioma determinado.



## Glosario

**diccionario de compresión.** En DB2 Universal Database para OS/390, diccionario que controla el proceso de compresión y descompresión. Este diccionario se crea a partir de los datos del espacio de tablas o de la partición de espacio de tablas.

**diccionario de datos.** Depósito de información acerca de los programas de aplicación, las bases de datos, los modelos de datos lógicos, los usuarios y las autorizaciones de una organización. Un diccionario de datos puede ser manual o automático.

**dimensión.** Categoría de datos, por ejemplo hora, cuentas, productos o mercados. Los elementos de una dimensión se denominan miembros. Las dimensiones ofrecen un modo simple muy conciso de organizar y seleccionar datos para la recuperación, la exploración y el análisis. Las dimensiones también representan el nivel de consolidación más alto en un esquema de base de datos multidimensional. Véase también “dimensión comercial”, “análisis multidimensional” en la página 1528 y “tabla de dimensión” en la página 1615.

**dimensión comercial.** Categoría de datos, por ejemplos productos o periodos de tiempo, que una organización puede querer analizar. Véase también “dimensión” y “análisis multidimensional” en la página 1528.

**dirección conocida públicamente.** Dirección que se utiliza para identificar de forma exclusiva un nodo determinado en la red para establecer conexiones entre nodos. La dirección conocida públicamente es una combinación de la dirección de red y del puerto utilizado en el nodo lógico.

**dirección de byte relativa (RBA).** En un entorno OS/390, desplazamiento de un registro de datos o intervalo de control respecto al principio del espacio de almacenamiento asignado al conjunto de datos o archivo al cual pertenece el registro.

**dirección de red.** Identificador de un nodo en una red.

**dirección IP.** Valor de 4 bytes que identifica de forma exclusiva un sistema principal TCP/IP.

**directorio.** Base de datos del sistema DB2 Universal Database para OS/390 que contiene objetos internos tales como ejemplo descriptores de bases de datos y tablas esquemáticas de cursor.

**directorio actual de trabajo.** Directorio por omisión de un proceso a partir del cual se determinan todas las vías de acceso relativas.

**directorio de bases de datos.** Directorio que contiene información de acceso a bases de datos para todas las bases de datos a las que se puede conectar un cliente. Véase “directorio de nodos”.

**directorio de bases de datos del sistema.** Directorio que contiene entradas para cada base de datos a la que se puede tener acceso utilizando el gestor de bases de datos. Este directorio se crea cuando se crea o cataloga la primera base de datos en el sistema. Véase también “directorio de bases de datos locales”.

**directorio de bases de datos locales.** Directorio en el que reside físicamente una base de datos. Las bases de datos que se visualizan en el directorio de bases de datos locales están ubicadas en el mismo nodo que el directorio de bases de datos del sistema (“directorio de bases de datos del sistema”).

**directorio de nodos.** Directorio que contiene la información necesaria para establecer comunicaciones entre una estación de trabajo cliente y todos los servidores de base de datos aplicables.

**directorio de red distribuido.** Véase “base de datos de directorios distribuidos” en la página 1532.



**directorio de servicios de conexión de base de datos (database connection services - DCS).** Directorio que contiene entradas correspondientes a bases de datos remotas y el peticionario de aplicaciones utilizado para acceder a ellas.

**distintivo.** Opción de precompilador que identifica sentencias de SQL de aplicaciones que no cumplen los criterios de validación seleccionados (por ejemplo, la norma SQL92 de ISO/ANSI).

**distintivo de indicador nulo.** Carácter de un byte que está contenido en una columna de “indicador nulo” en la página 1571 de un archivo ASCII no delimitado. Cuando el proceso de carga examina cada fila de datos, el distintivo de indicador nulo indica si los datos de la columna definida por las posiciones inicial y final son nulos.

**DLC.** Véase “control de enlace de datos” en la página 1545.

**DLU.** Véase “unidad lógica dependiente” en la página 1621.

**DML.** Véase “lenguaje de manipulación de datos” en la página 1576.

**DNS.** Véase “sistema de nombres de dominio” en la página 1610.

**dominio.** Un dominio forma parte de una red y se administra como una unidad con protocolo común.

**DRDA.** Véase “Arquitectura de Bases de Datos Relacionales Distribuidas” en la página 1531.

**drenaje físico.** En DB2 Universal Database para OS/390, drenaje sobre un índice completo sin particionamiento.

**drenaje lógico.** En DB2 Universal Database para OS/390, drenaje sobre una partición lógica de un índice sin particionamiento.

**drenar.** En DB2 Universal Database para OS/390, adquirir un recurso bloqueado desactivando el acceso a dicho objeto. Véase también “reclamación” en la página 1599.

**DSN.** Véase “nombre de subsistema por omisión” en la página 1585.

**dudoso.** Estado de una unidad de recuperación. Si el gestor de bases de datos falla después de finalizar la fase 1 del proceso de confirmación y antes de iniciar la fase 2, sólo el coordinador de confirmación sabe si una unidad de recuperación individual debe confirmarse o retrotraerse. En un reinicio de emergencia, si el gestor de bases de datos carece de la información que necesita para tomar esta decisión, el estado de la unidad de recuperación será *dudoso* hasta que el gestor de bases de datos obtenga esta información del coordinador. Puede haber más de una unidad de recuperación en estado dudoso durante el reinicio.

**DUOW.** Véase “unidad de trabajo distribuida” en la página 1621.

**duplicación.** Proceso de mantener un conjunto definido de datos en más de un lugar. Supone copiar cambios designados de un lugar a otro (fuente y destino, respectivamente) y sincronizar los datos de ambas ubicaciones.

**duplicación de agrupaciones de almacenamientos intermedios de grupo.** En un entorno OS/390, capacidad de escribir datos en dos instancias de una agrupación de almacenamientos intermedios de grupo: una *agrupación primaria de almacenamientos intermedios de grupo* y una *agrupación secundaria de*

## Glosario

*almacenamientos intermedios de grupo.* En las publicaciones sobre OS/390, estas instancias se denominan estructura "antigua" (para la primaria) y estructura "nueva" (para la secundaria).

**duplicado.** Tipo de tabla destino que se puede actualizar localmente y que recibe actualizaciones de una tabla de usuario a través de una definición de suscripción. Puede ser una fuente para actualizar la tabla de usuario o tablas destino de sólo lectura.

**duplicado de fila.** En el proceso de duplicación de datos de DB2, tipo de duplicado con actualización en cualquier lugar mantenida por DataPropagator para Microsoft Jet sin semánticas de transacción.

**duración.** En SQL, número que representa un intervalo de tiempo. Véase "duración de fecha", "duración etiquetada" y "duración de hora".

**duración de fecha.** Valor de tipo DECIMAL(8,0) que representa un número de años, meses y días.

**duración de hora.** Valor DECIMAL(6,0) que representa un número de horas, minutos y segundos.

**duración de indicación de fecha y hora.** Valor DECIMAL(20,6) que representa un número de años, meses, días, horas, minutos, segundos y microsegundos.

**duración del bloqueo.** Intervalo durante el cual se mantiene un bloqueo de DB2 Universal Database para OS/390. Por ejemplo, los bloqueos en los LOB se realizan cuando son necesarios y se liberan generalmente en la confirmación.

**duración etiquetada.** Número que representa una duración de años, meses, días, horas, minutos, segundos o microsegundos.

## E

**EBCDIC.** Juego de caracteres codificado de 256 caracteres de 8 bits, utilizado normalmente en servidores S/390 y AS/400. Véase también "ASCII" en la página 1531 y "Unicode" en la página 1621.

**edición.** Véase "edición de paso".

**edición de enlaces de memoria cruzada.** En un entorno OS/390, método para invocar un programa situado en un espacio de direcciones diferente. La invocación es síncrona respecto al emisor de la llamada.

**edición de paso.** En Centro de depósito de datos, visión de los datos existentes en una fuente de depósito en un momento determinado.

**editar enlaces.** En DB2 Universal Database para OS/390, acción de crear un programa cargable utilizando un editor de enlaces.

**editor de enlaces.** Programa informático para crear módulos de carga a partir de uno o más módulos objeto mediante la resolución de referencias cruzadas entre los módulos y, si es necesario, el ajuste de direcciones.

**EID (event identifier).** Identificador de suceso.

**elemento de código.** Patrón de bits exclusivo que representa un carácter en una página de códigos.

**elemento de datos.** Estructura de datos utilizada por el supervisor del sistema para almacenar información relacionada con el estado del sistema de base de datos. Por ejemplo, contadores, marcadores, información e indicaciones de fecha y hora.

**EN.** Véase “nodo final” en la página 1582.

**en cancelación anómala.** Estado de una unidad de recuperación. Si DB2 Universal Database para OS/390 falla después de que empiece a retrotraerse una unidad de recuperación, pero antes de que el proceso finalice, DB2 Universal Database para OS/390 seguirá restituyendo los cambios durante el reinicio.

**enclavamiento.** Mecanismo interno de DB2 Universal Database para OS/390 que sirve para controlar sucesos simultáneos o el uso de recursos del sistema.

**enclave.** En Language Environment (el cual es utilizado por DB2 Universal Database para OS/390), conjunto independiente de rutinas, una de las cuales está designada como rutina principal. Un enclave es similar a un programa o unidad de ejecución.

**en confirmación.** Estado de una unidad de recuperación. Si el DB2 Universal Database para OS/390 falla después de empezar el proceso de confirmación de dos fases, cuando reorganiza, “sabe” que los cambios realizados en los datos son coherentes.

**enlace.** Proceso mediante el cual la salida del compilador SQL se convierte en una estructura de control utilizable, denominada normalmente plan de acceso, plan de aplicación o paquete. Durante este proceso se seleccionan vías de acceso a los datos y se efectúa una comprobación de autorización. Véase también “reenlazar” en la página 1601, “reenlace automático” en la página 1601, “enlace dinámico”, “enlace incremental”, “enlace estático”.

**enlace dinámico.** Proceso mediante el cual las sentencias de SQL se enlazan cuando se entran. Véase “enlace”. Véase también “enlace estático”.

**enlace estático.** Proceso mediante el cual las sentencias de SQL se enlazan una vez precompiladas. Todas las sentencias de SQL estático se preparan al mismo tiempo para su ejecución. Véase “enlace”. Véase también “enlace dinámico”.

**enlace incremental.** Proceso mediante el cual las sentencias de SQL se enlazan durante la ejecución de un proceso de aplicación, porque no se pudieron enlazar durante el proceso de enlace y se especificó VALIDATE(RUN). Véase “enlace”.

**entero binario.** Tipo de datos básico que se puede clasificar adicionalmente como entero pequeño o entero grande.

**entidad.** (1) En una base de datos relacional, las entidades se representan como tablas. Una base de datos incluye información acerca de las entidades de una organización o empresa y las relaciones entre ellas. Una entidad es una persona, un objeto o un concepto sobre el cual se desea almacenar información. (2) Unidad de datos que se puede clasificar y que puede tener relaciones establecidas con otras entidades de dicha base de datos.

**entorno de aplicación del WLM.** Atributo del Gestor de Cargas de Trabajo de MVS que está asociado a uno o más procedimientos almacenados. El entorno de aplicación del WLM determina el espacio de direcciones en el que se ejecuta un determinado procedimiento almacenado de DB2 Universal Database para OS/390.

## Glosario

**entorno nacional.** En DB2 Universal Database para OS/390, definición de un subconjunto de un entorno de usuario que combina caracteres definidos para un idioma y país determinados, y un CCSID.

**entrada de recuperación residual (RRE).** Unidad de recuperación de la que DB2 puede estar dudoso. Por ejemplo, en un entorno OS/390 con IMS, IMS crea una lista de entradas de recuperación residuales.

**envío de función.** Envío de las subsecciones de una petición a la partición de base de datos específica que contiene los datos aplicables.

**en vuelo.** Estado de una unidad de recuperación. Si DB2 Universal Database para OS/390 falla antes de que su unidad de recuperación finalice la fase 1 del proceso de confirmación, simplemente cancela las actualizaciones de su unidad de recuperación cuando rearranca. Estas unidades de recuperación se denominan *en vuelo*.

**EOM (end of memory).** Fin de memoria.

**EOT (end of task).** Fin de tarea.

**E/S asíncrona.** Proceso no secuencial de peticiones de lectura y grabación en múltiples discos.

**E/S basada en bloque.** Método del gestor de bases de datos para leer páginas de datos contiguas de disco en partes contiguas de memoria. Véase también “lectura diseminada” en la página 1576.

**escala.** Número de dígitos de la parte fraccionaria de un número.

**escenario de verificación de instalación.** Secuencia de operaciones que realiza las principales funciones de DB2 Universal Database y prueba si DB2 se ha instalado correctamente.

**escribir a operador (WTO).** Servicio opcional codificado por el usuario que permite escribir un mensaje para el operador de la consola del sistema para informarle de errores y condiciones poco habituales del sistema que pueden necesitar corrección.

**ESDS.** En un entorno OS/390, archivo en secuencia de entrada.

**E/S en paralelo.** Proceso de leer o escribir en dos o más dispositivos de E/S al mismo tiempo para reducir el tiempo de respuesta.

**ESMT.** Véase “tabla de módulo de subsistema externa” en la página 1615.

**E/S orientable.** Véase “lectura diseminada” en la página 1576.

**espacio de datos.** En DB2 Universal Database para OS/390, espacio, cuyo tamaño está en un rango de 0 bytes a 2 gigabytes menos 1 byte, de direcciones de almacenamiento virtual contiguo que un programa puede manipular directamente. A diferencia de un espacio de direcciones, un espacio de datos sólo puede contener datos; no contiene áreas comunes, datos del sistema ni programas.

**espacio de direcciones.** Memoria real utilizada por un programa activo. Véase también “agrupación de almacenamientos intermedios” en la página 1526.

**espacio de direcciones aliado.** Área de almacenamiento que es externa a DB2 Universal Database para OS/390 y está conectada a él. Un espacio de direcciones aliado puede solicitar servicios de DB2 Universal Database para OS/390.

**espacio de direcciones inicial.** En un entorno OS/390 área de almacenamiento que OS/390 reconoce actualmente como *despachado*.

**espacio de índice.** En DB2 Universal Database para OS/390, conjunto de páginas utilizado para almacenar las entradas de un índice.

**espacio de tabla grande.** Espacio de tabla que sólo puede almacenar datos de serie larga, de objeto grande (LOB) o de índice.

**espacio de tablas.** (1) Concepto abstracto que designa un conjunto de contenedores en los que se almacenan objetos de base de datos. Un espacio de tablas proporciona un nivel de direccionamiento indirecto entre una base de datos y las tablas contenidas en ella. (2) Un espacio de tablas tiene espacio en dispositivos de almacenamiento de soporte asignados a él. (3) Un espacio de tablas tiene tablas creadas en él. Estas tablas ocupan espacio en los contenedores que pertenecen al espacio de tablas. Las porciones de datos, índice, campo largo y LOB de una tabla pueden almacenarse en el mismo espacio de tablas o pueden dividirse individualmente en espacios de tablas independientes. (4) En DB2 Universal Database para OS/390, conjunto de páginas utilizado para almacenar registros en una o más tablas.

**espacio de tablas base.** En DB2 Universal Database para OS/390, espacio de tablas que contiene "tablas base".

**espacio de tablas con espacio gestionado por el sistema (SMS).** Espacio de tablas cuyo espacio está gestionado por el sistema operativo. Este modelo de almacenamiento está basado en archivos creados en subdirectorios y está gestionado por el sistema de archivos. Véase también "espacio de tablas con espacio gestionado por la base de datos".

**espacio de tablas con espacio gestionado por la base de datos (DMS).** Espacio de tablas cuyo espacio está gestionado por la base de datos. Véase también "espacio de tablas con espacio gestionado por el sistema".

**espacio de tablas de LOB.** En DB2 Universal Database para OS/390, espacio de tablas que contiene todos los datos para una columna de LOB determinada de la tabla base asociada.

**espacio de tablas DMS.** Véase "espacio de tablas con espacio gestionado por la base de datos".

**espacio de tablas habilitado para direccionabilidad ampliada.** En DB2 Universal Database para OS/390, espacio de tablas o espacio de índice que está habilitado para la direccionabilidad ampliada y que contiene particiones individuales (o partes, para espacios de tablas LOB) que tienen más de 4 gigabytes de tamaño.

**espacio de tablas largo.** Véase "espacio de tabla grande".

**espacio de tablas padre.** En DB2 Universal Database para OS/390, espacio de tablas que contiene una tabla padre. Un espacio de tablas que contiene un objeto dependiente de esa tabla es un espacio de tablas dependiente.

**espacio de tablas particionado.** En un entorno OS/390, espacio de tablas subdividido en partes (de acuerdo con el rango de la clave de índice), cada una de las cuales puede ser procesada independientemente mediante programas de utilidad.

**espacio de tablas regular.** Espacio de tablas que puede almacenar cualquier dato no temporal.

## Glosario

**espacio de tablas segmentado.** En DB2 Universal Database para OS/390, espacio de tablas dividido en grupos de igual tamaño de páginas llamados segmentos. Los segmentos se asignan a tablas de modo que las filas de tablas diferentes no se almacenan nunca en el mismo segmento.

**espacio de tablas simple.** En DB2 Universal Database para OS/390, espacio de tablas que no está particionado ni segmentado.

**espacio de tablas SMS.** Véase “espacio de tablas con espacio gestionado por el sistema” en la página 1559.

**espacio de tablas temporales.** Espacio de tablas que sólo puede almacenar tablas temporales.

**espacio libre.** Cantidad total de espacio no utilizado de una página. El espacio que no se utiliza para almacenar registros ni información de control es espacio libre.

**especificación de índice.** En un sistema de bases de datos federado, conjunto de metadatos referentes a una tabla fuente de datos. Estos metadatos constan de información que habitualmente se encuentra en una definición de índice, por ejemplo, qué columna o columnas deben examinarse para recuperar información rápidamente. El usuario podría proporcionar estos metadatos al servidor federado si la tabla no tiene ningún índice o si tiene un índice que el servidor federado no conoce. La finalidad de los metadatos es facilitar la recuperación de los datos de la tabla.

**esquema.** (1) Conjunto de objetos de bases de datos, por ejemplo tablas, vistas, índices o desencadenantes, que definen una base de datos. Un esquema de base de datos proporciona una clasificación lógica de objetos de base de datos. (2) En DB2 Universal Database para OS/390, agrupación lógica de funciones definidas por el usuario, tipos diferenciados, desencadenantes y procedimientos almacenados. Cuando se crea un objeto de uno de estos tipos, se asigna a un esquema, que está determinado por el nombre del objeto. (3) En Centro de depósito de datos, conjunto de tablas destino de depósito y las relaciones existentes entre las columnas de esas tablas; las tablas destino pueden proceder de uno o más destinos de depósito.

**esquema de codificación.** Conjunto de reglas para representar datos de tipo carácter.

**esquema en estrella.** Tipo de esquema de base de datos relacional utilizado por el Kit de iniciación de OLAP; a menudo se crea en el Centro de depósito de datos.

**estabilidad del cursor (CS).** Nivel de aislamiento que bloquea cualquier fila a la que tiene acceso una transacción de una aplicación mientras el cursor está situado en la fila. El bloqueo permanece en vigor hasta que se lee la fila siguiente o finaliza la transacción. Si se cambian datos de una fila, el bloqueo se mantiene hasta que el cambio se confirma en la base de datos. Véase también “estabilidad de lectura”, “lectura repetible” en la página 1576 y “lectura no confirmada (UR)” en la página 1576.

**estabilidad de lectura (RS).** Nivel de aislamiento que bloquea solamente las filas que una aplicación recupera dentro de una transacción. La estabilidad de lectura asegura que una fila que se está leyendo durante una transacción no sea cambiada por otros procesos de aplicación hasta que haya finalizado la transacción y que una fila cambiada por otro proceso de aplicación no se lea hasta que ese proceso confirme el cambio. La estabilidad de lectura permite más simultaneidad que la lectura repetible y menos simultaneidad que la estabilidad de cursor. Véase también “estabilidad del cursor”, “lectura repetible” en la página 1576 y “lectura no confirmada (UR)” en la página 1576.

**estadísticas explicadas.** Estadísticas para un objeto de base de datos al que se hizo referencia en una sentencia de SQL cuando se explicó la sentencia.

**estadísticas modeladas.** Estadísticas para un objeto de la base de datos al que se puede hacer referencia o no en una sentencia de SQL, pero que sin embargo existe actualmente en un modelo de explicación. El objeto puede existir o no actualmente en la base de datos.

**estado.** En Centro de depósito de datos, fase de trabajo en curso de un paso de proceso, tal como "planificado", "llenando con datos" o "satisfactorio".

**estado de miembro.** En DB2 Universal Database para OS/390, estado del miembro (subsistema) DB2 del grupo de compartimiento de datos.

**estado de miembro anómalo.** En DB2 Universal Database para OS/390, estado de un miembro perteneciente a un grupo de compartimiento de datos. Cuando un miembro falla, XCF registra de forma permanente el estado de miembro anómalo. Este estado suele significar que la tarea, espacio de direcciones o sistema MVS del miembro han finalizado antes de que el estado cambiara de activo a inmovilizado.

**estado de miembro inmovilizado.** En DB2 Universal Database para OS/390, estado de un miembro perteneciente a un grupo de compartimiento de datos. Un miembro activo pasa a ser un miembro inmovilizado cuando un mandato STOP DB2 surte efecto sin ningún error. Si la tarea, espacio de direcciones o sistema OS/390 del miembro falla antes de que surta efecto el mandato, el estado del miembro es anómalo.

**estrategia de particionamiento basado en valor de clave.** Método para asignar filas de una tabla a particiones de base de datos. Las filas se asignan basándose en los valores de las columnas de clave de particionamiento.

**estructura.** Nombre que hace referencia de forma colectiva a diferentes tipos de objetos de DB2, por ejemplo tablas, bases de datos, vistas, índices y espacios de tablas.

**estructura de antememoria.** Una estructura de recurso de acoplamiento que almacena datos que pueden estar disponibles para todos los miembros de un Parallel Sysplex. Un grupo de compartimiento de datos de DB2 Universal Database para OS/390 utiliza estructuras de antememoria como agrupación de almacenamiento intermedio de grupo ("agrupación de almacenamientos intermedios de grupo" en la página 1526).

**estructura de bloqueo.** En DB2 Universal Database para OS/390, estructura de datos del recurso de acoplamiento formada por una serie de entradas de bloqueo para dar soporte al bloqueo compartido y exclusivo para recursos lógicos.

**estructura de lenguaje principal.** En un programa de aplicación, estructura a la que hacen referencia sentencias de SQL intercaladas.

**estructura de lista.** En un entorno OS/390, estructura del recurso de acoplamiento que permite compartir y manejar datos como elementos de una cola.

**estructura referencial.** En DB2 Universal Database para OS/390, conjunto de tablas y relaciones que incluye como mínimo una tabla y, para cada tabla del conjunto, todas las relaciones en las que participa la tabla y todas las tablas con las que está relacionada.

**EUC.** Véase "Extended <tm trademark="UNIX" tmowner="X/Open Company Limited" tmtype="reg" tmclass="special">UNIX</tm> Code" en la página 1623.



## Glosario

**expansor de DB2.** Programa que permite al usuario almacenar y recuperar tipos de datos tales como imágenes, audio, vídeo y documentos complejos, además de los datos tradicionales de tipo numérico o de tipo carácter.

**explicar.** Capturar información detallada acerca del plan de acceso elegido por el compilador de SQL para resolver una sentencia de SQL. La información describe los criterios de decisión utilizados para elegir el plan de acceso.

**exportar.** Copiar datos de tablas del gestor de bases de datos a un archivo utilizando formatos tales como PC/IXF, DEL, WSF o ASC. Véase también “importar” en la página 1571.

**expresión.** Operando o conjunto de operadores de SQL que produce un valor individual.

**expresión CASE.** Expresión que permite seleccionar otra expresión basándose en la evaluación de una o más condiciones.

**expresión de tabla.** Expresión que crea una tabla resultante (temporal) a partir de una consulta simple. Por ejemplo, una expresión de tabla puede ser una consulta que seleccione todos los directores de diversos departamentos y además especifique que éstos deben tener más 15 años de experiencia y que deben encontrarse en el laboratorio de Toronto. Véase también “expresión de tabla común”.

**expresión de tabla anidada.** Selección completa en una cláusula FROM (entre paréntesis).

**expresión de tabla común.** Expresión que define una tabla resultante con un nombre (identificador calificado de SQL). Puede especificarse como un nombre de tabla en cualquier cláusula FROM de la selección completa que sigue a la cláusula WITH. Véase “expresión de tabla”.

**expresión de tabla común recurrente.** Expresión de tabla común que hace referencia a sí misma en una cláusula FROM de la selección completa. Las expresiones de tabla común recurrentes se utilizan para escribir consultas recurrentes.

**expresión de valor de fila.** En el entorno OS/390, lista de expresiones de valor entre paréntesis y separadas por coma.

**extensión.** Asignación de espacio, dentro de un contenedor de un espacio de tablas, a un objeto individual de base de datos. Esta asignación consta de varias páginas.

## F

**factor de filtro.** En DB2 Universal Database para OS/390, número comprendido entre cero y un valor correspondiente a la proporción de filas de una tabla para las cuales un predicado es verdadero. Se dice que dichas filas se califican mediante ese predicado. Los factores de filtro afectan a la elección de las vías de acceso calculando el número de filas calificadas por un conjunto de predicados.

**familia de funciones.** Conjunto de funciones con el mismo nombre de función. El contexto determina si el uso hace referencia a un conjunto de funciones dentro de un esquema determinado o a todas las funciones aplicables que tienen el mismo nombre dentro de la vía de acceso de la función actual.

**familia de vías de acceso de función.** Todas las funciones del nombre especificado de todos los esquemas identificados (o utilizados por omisión) en la vía de acceso de función del usuario.



**fase confirmada.** En DB2 Universal Database, segunda fase del proceso de actualización de múltiples ubicaciones que solicita a todos los participantes que confirmen los efectos de la unidad lógica de trabajo.

**fecha.** Valor compuesto de tres partes que indica un día, un mes y un año. Por ejemplo, AAAA-MM-DD.

**fetch.** La sentencia FETCH coloca un cursor en la fila siguiente de su tabla resultante y asigna los valores de dicha fila a las variables del lenguaje principal.

**fila.** Componente horizontal de una tabla que consta de una secuencia de valores, uno para cada columna de la tabla.

**fila de autorreferencia.** Fila que es padre de sí misma.

**fila dependiente.** Fila que contiene una clave foránea que coincide con el valor de una clave padre de la fila padre. El valor de la clave foránea representa una referencia desde la fila dependiente a la fila padre.

**fila descendiente.** Fila dependiente de otra fila o que es descendiente de una fila dependiente.

**fila fantasma.** Fila de tabla que pueden leer los procesos de aplicación que se están ejecutando con cualquier nivel de aislamiento excepto el de lectura repetible. Cuando un proceso de aplicación emite la misma consulta varias veces dentro de una sola unidad de trabajo, pueden aparecer filas adicionales entre las consultas debido a los datos que están insertando y confirmando los procesos de aplicación que se ejecutan simultáneamente.

**fila padre.** Fila que tiene como mínimo una fila dependiente.

**físicamente completado.** En DB2 Universal Database para OS/390, estado en el que el proceso de copia concurrente ha finalizado y se ha creado el archivo de salida.

**formato ASCII (ASC) no delimitado.** Formato de archivo que se utiliza para importar datos. ASCII no delimitado es un archivo ASCII secuencial con delimitadores de fila que se utiliza para el intercambio de datos con cualquier producto ASCII.

**fragmentación.** Separación del índice en partes independientes como resultado de las inserciones y supresiones realizadas en el índice.

**fuelle.** En Centro de depósito de datos, tabla, vista o archivo que se utiliza como datos de entrada de un paso de proceso. Véase también “destino” en la página 1553.

**fuelle de depósito.** Subconjunto de tablas y vistas de una base de datos individual, o conjunto de archivos, que se han definido para el Centro de depósito de datos.

**fuelle de duplicación.** Tabla o vista de base de datos que puede aceptar peticiones de copia y es la tabla fuente de un conjunto de suscripción. Véase también “conjunto de suscripción” en la página 1544.

**función.** Correlación, en forma de programa (cuerpo de la función), que se puede invocar por medio de cero o más valores de entrada (argumentos) para obtener un valor individual (resultado). Las funciones puede estar definidas por el usuario, incorporadas o generadas por DB2. Véase también “función de columna” en la página 1564 y “función escalar” en la página 1564.

## Glosario

**función agregada.** Sinónimo de “función de columna”.

**función con fuente.** Función implementada por otra función incorporada o definida por el usuario que ya se conoce en el gestor de bases de datos. Esta función puede ser una función escalar o una función de columna (de agregación); devuelve un único valor de un conjunto de valores (por ejemplo, MAX o AVG). Véase también “función externa”, “función definida por el usuario”, “función incorporada” y “función SQL” en la página 1565.

**función de acceso.** Función, proporcionada por el usuario, que convierte el tipo de datos del texto almacenado en una columna a un tipo que puede ser procesado por el Text Extender.

**función de base de datos.** Relación entre un conjunto de datos de entrada y un conjunto de valores de resultado. Véase también “función incorporada” y “función definida por el usuario”.

**función de columna.** (1) Operación que obtiene su resultado utilizando valores de una o más filas. (2) Función que realiza un cálculo en un conjunto de valores en lugar de hacerlo en un valor individual. Sinónimo de “función agregada”. Véase también “función escalar” y “función de tabla”.

**función de conversión del tipo de datos.** Función que se utiliza para convertir instancias de un tipo de datos fuente en instancias de un tipo de datos de destino diferente. En general, una función de conversión tiene el nombre del tipo de datos de destino y tiene un solo argumento cuyo tipo es el tipo de datos fuente. El tipo devuelto es el tipo de datos de destino.

**función de fila.** Función que devuelve una fila de valores y debe definirse como una función de SQL.

**función definida por el usuario (UDF).** Función que se define en DB2 utilizando la sentencia CREATE FUNCTION y a la que, a partir de ese momento, se puede hacer referencia en las sentencias de SQL. Una función definida por el usuario puede ser una función externa (“función SQL” en la página 1565) o una función de SQL (“función SQL” en la página 1565). Véase también “función incorporada”.

**función de tabla.** Función que recibe un conjunto de argumentos y devuelve una tabla a la sentencia de SQL que hace referencia a la función. Sólo se puede hacer referencia a una función de tabla en la cláusula FROM de una subselección. Véase también “función de columna” y “función escalar”.

**función determinista.** Función definida por el usuario cuyo resultado depende únicamente de los valores de los argumentos de entrada. Las invocaciones sucesivas con los mismos valores de argumentos producen siempre los mismos resultados.

**función escalar.** Operación de SQL que produce un solo valor a partir de otro valor y se expresa como un nombre de función seguido de una lista de argumentos encerrados entre paréntesis. Véase también “función de columna” y “función de tabla”.

**función externa.** Función cuyo cuerpo está escrito en un lenguaje de programación que toma valores de argumento escalares y produce un resultado escalar para cada invocación. Véase también “función con fuente” y “función incorporada”.

**función incorporada.** Función de SQL proporcionada por DB2 y que aparece en el esquema SYSIBM. Véase también “función definida por el usuario”.

**función no determinista.** En DB2 Universal Database para OS/390, función definida por el usuario cuyo resultado no depende únicamente de los valores de los argumentos de entrada. Invocaciones sucesivas con los mismos valores de argumento pueden producir una respuesta diferente. Este tipo de

función se denomina a veces función variable (“función variable”). Una función determinista (“función determinista” en la página 1564) produce siempre el mismo resultado para la misma entrada.

**función no variable.** Sinónimo de “función determinista” en la página 1564. Véase también “función variable”.

**función SQL.** Función definida por el usuario en la que la sentencia CREATE FUNCTION contiene el código fuente. El código fuente es una expresión individual de SQL que se evalúa en un valor individual. La función de SQL definida por el usuario sólo puede devolver un parámetro.

**función variable.** Función definida por el usuario cuyo resultado depende de los valores de los parámetros de entrada así como de otros factores. Invocaciones sucesivas con los mismos valores de parámetro pueden producir resultados diferentes. Véase también “función no determinista” en la página 1564.

**fusionar.** Método para actualizar e insertar contenido nuevo en una tabla.

## G

**GBP (group buffer pool).** Véase “agrupación de almacenamientos intermedios de grupo” en la página 1526.

**generador de archivo de respuestas.** Este programa de utilidad crea un archivo de respuestas de un producto DB2 existente instalado y configurado. Puede utilizar el archivo de respuestas generado para volver a crear la configuración exacta en otras máquinas.

**generador de declaraciones (declarations generator - DCLGEN).** Subcomponente de DB2 Universal Database para OS/390 que genera declaraciones de tablas SQL y declaraciones de estructuras de datos COBOL, C o PL/I que se ajustan a la tabla. Las declaraciones se generan a partir de información contenida en catálogos del sistema de DB2 Universal Database para OS/390. DCLGEN también es un submandato de DSN.

**Generador de Procedimientos Almacenados.** Herramienta para crear procedimientos almacenados en servidores DB2 locales y remotos, modificar y volver a crear procedimientos almacenados existentes y probar y depurar su ejecución utilizando una interfaz gráfica. Esta herramienta es autónoma y también se puede acceder a ella desde diversos entornos de desarrollo integrados.

**Gestor de antememoria.** En Net.Data, programa que gestiona una antememoria de una estación de trabajo individual. El Gestor de Antememoria puede gestionar varias antememorias.

**gestor de bases de datos.** Programa de software que gestiona datos proporcionando los servicios de control centralizado, independencia de datos y estructuras físicas complejas para lograr acceso, integridad, recuperación, control de simultaneidad, privacidad y seguridad eficaces.

**Gestor de Catálogos de Información.** Aplicación para organizar, mantener, encontrar y utilizar información de una empresa.

**gestor de comunicaciones rápido (fast communication manager, FCM).** Grupo de funciones que proporcionan soporte de comunicaciones entre nodos.

## Glosario

**Gestor de conexiones.** En Net.Data, archivo ejecutable, dtwcm que se necesita para soportar Live Connection.

**gestor de transacciones.** Función que asigna identificadores a transacciones, supervisa su progreso y asegura la realización de las transacciones y la recuperación de errores.

**GIMSMP.** En un entorno OS/390, nombre del módulo de carga correspondiente a System Modification Program/Extended, herramienta básica utilizada para instalar, modificar y controlar cambios en sistemas de programación.

**GIS.** Véase “sistema de información geográfica” en la página 1610.

**grabación.** Información procedente de instantáneas de rendimiento que puede visualizarse posteriormente.

**grabación diferida.** En DB2 Universal Database para OS/390, hace referencia al proceso de grabar asíncronamente en disco páginas de datos cambiadas.

**grado de paralelismo.** Número de operaciones ejecutadas concurrentemente que se inician para procesar una petición.

**gran objeto binario (BLOB).** Secuencia de bytes con un tamaño en un rango de 0 bytes a 2 gigabytes menos 1 byte. Esta secuencia no tiene ninguna página de códigos ni juego de caracteres asociados. Los BLOB pueden contener datos de imagen, audio y vídeo. Véase también “gran objeto de caracteres” y “gran objeto de caracteres de doble byte”.

**gran objeto de caracteres (CLOB).** Secuencia de caracteres (de un solo byte y/o de múltiples bytes) con un tamaño en un rango de 0 bytes a 2 gigabytes menos 1 byte. En general, los valores de los grandes objetos de caracteres se utilizan siempre que una serie de caracteres puede exceder los límites del tipo VARCHAR. También se denomina serie de gran objeto de caracteres. Véase también “gran objeto binario” y “gran objeto de caracteres de doble byte”.

**gran objeto de caracteres de doble byte (DBCLOB).** Secuencia de caracteres de doble byte, con un tamaño en un rango de 0 bytes a 2 gigabytes menos 1 byte. Tipo de datos que se puede utilizar para almacenar objetos grandes de texto de doble byte. Este tipo de serie de caracteres tiene siempre una página de códigos asociada. Véase también “gran objeto binario” y “gran objeto de caracteres”.

**granularidad del desencadenante.** Característica de un desencadenante, que determina si éste se activa una vez para la sentencia de SQL desencadenante o una vez para cada fila que la sentencia de SQL modifica.

**grupo.** (1) Organización lógica de usuarios cuyos ID se establecen de acuerdo con la actividad o la autorización de acceso a recursos. (2) En un entorno de satélite, conjunto de satélites que comparten características, por ejemplo la configuración de la base de datos y la aplicación que se ejecuta en el satélite.

**grupo de almacenamiento.** Conjunto de discos con nombre en el que se pueden almacenar datos de DB2 Universal Database para OS/390.

**grupo de compartimiento de datos.** Conjunto de uno o más subsistemas de DB2 Universal Database para OS/390 que acceden directamente y modifican los mismos datos sin alterar su integridad.

**grupo de nodos.** Grupo, con un nombre asignado, de una o más particiones de base de datos.

**grupo de programas de depósito.** En Centro de depósito de datos, contenedor (carpeta) que contiene objetos de programa.

**grupo de repetición.** Situación en la que una entidad incluye múltiples atributos que son inherentemente iguales. La presencia de un grupo de repetición viola el requisito de la primera forma normal. En una entidad que satisface el requisito de la primera forma normal, cada atributo es independiente y exclusivo en su significado y su nombre. Véase también “normalización” en la página 1585.

**grupo paralelo.** En un entorno OS/390, conjunto de operaciones consecutivas que se ejecutan en paralelo y tienen el mismo número de tareas paralelas.

**GTF.** Véase “recurso de rastreo generalizado” en la página 1601.

**guión gráfico.** Resumen visual de una película de vídeo. El Video Extender incluye características que permiten identificar y almacenar fotogramas de vídeo que son representativos de las secuencias de una película de vídeo. Estos fotogramas representativos se pueden utilizar para crear un guión gráfico.

**GWAPI.** API del servidor Web Domino Go.

## H

**habilitador de duplicación móvil.** Programa de duplicación que inicia la modalidad de duplicación móvil en el cliente móvil.

**habilitar.** (1) Preparar una base de datos, una tabla de texto o una columna de texto para que la utilice Text Extender. (2) Encender o activar.

**hebra.** (1) Estructura del gestor de bases de datos que describe la conexión de una aplicación, rastrea su proceso, procesa las funciones de recursos y delimita su posibilidad de acceso a los recursos y servicios de DB2 Universal Database para OS/390. (2) En algunos sistemas operativos, unidad más pequeña de operación que debe efectuarse en un proceso. (3) Estructura de base de datos que describe la conexión de una aplicación, rastrea su proceso, procesa las funciones de recursos y delimita su posibilidad de acceso a los recursos y servicios del gestor de bases de datos. La mayoría de las funciones de DB2 Universal Database para OS/390 se ejecutan bajo una estructura de hebras. Véase también “hebra aliada” y “hebra de acceso a base de datos”.

**hebra aliada.** Hebra que se origina en el subsistema local DB2 Universal Database para OS/390 y que puede acceder a los datos de un subsistema remoto DB2 Universal Database para OS/390. Véase también “hebra”.

**hebra de acceso a base de datos.** En DB2 Universal Database para OS/390, hebra que accede a datos del sistema local por cuenta de un subsistema remoto. Véase también “hebra aliada”.

**herencia.** Transferencia de recursos o atributos de clase desde una clase padre a una clase hija, situada en un nivel inferior de la jerarquía de clases.

**herramienta DataJoiner Replication Administration (DJRA).** Herramienta de administración de bases de datos que se puede utilizar para realizar diversas tareas de administración de duplicación. A diferencia del Centro de Control, la herramienta DJRA se puede utilizar para gestionar la duplicación para bases de datos que no son de IBM. Véase también “Centro de control” en la página 1538.

## Glosario

**herramienta DJRA.** Herramienta de administración de bases de datos que se puede utilizar para realizar diversas tareas de administración de duplicación. A diferencia del Centro de Control, la herramienta DJRA se puede utilizar para gestionar la duplicación para bases de datos que no sean de IBM. Véase también “Centro de control” en la página 1538.

**hiperespacio.** En un entorno OS/390, espacio, cuyo tamaño está en un rango de 0 bytes a 2 gigabytes menos 1 byte, de direcciones de almacenamiento virtual contiguo que un programa puede utilizar como almacenamiento intermedio. Al igual que un espacio de datos, un hiperespacio puede contener datos de usuario; no contiene áreas comunes ni datos del sistema. A diferencia de un espacio de direcciones o de un espacio de datos, los datos de un hiperespacio no son directamente direccionables. Para manejar los datos de un hiperespacio, el usuario debe llevar los datos al espacio de direcciones en forma de bloques de 4 KB.

**hora.** Valor de tres partes que indica la hora en horas, minutos y segundos.

**hora de activación del desencadenante.** Indicación en una definición de desencadenante de si el desencadenante debe activarse antes o después del suceso desencadenado.

**Hora universal coordinada (UTC).** Sinónimo de Hora Media de Greenwich.

**HSM (hierarchical storage manager).** En un entorno OS/390, gestor de almacenamiento jerárquico.

**HTML.** Véase “Hypertext Markup Language”.

**hueco.** Especifica que el cursor tiene sensibilidad a los cambios efectuados en la base de datos después de que se materialice la tabla resultante. El cursor siempre es sensible a las actualizaciones y supresiones efectuadas utilizando el cursor (es decir, actualizaciones y supresiones posicionadas utilizando el mismo cursor). Cuando el valor actual de una fila ya no satisface la sentencia-select o el nombre-sentencia, dicha fila deja de estar visible mediante el cursor. Cuando una fila de la tabla resultante se suprime de la tabla base subyacente, la fila deja de estar visible mediante el cursor. La sensibilidad del cursor a los cambios realizados fuera de este cursor depende del hecho de que se utilice la sentencia SENSITIVE FETCH o la sentencia INSENSITIVE FETCH. Véase también “hueco de supresión” y “hueco de actualización”.

**hueco.** En el proceso de duplicación de datos de DB2, situación en la que el programa Capture no puede leer un rango de registros de diario o de archivo de anotaciones, por lo que pueden perderse datos de cambios.

**hueco de actualización.** Posibilidad para otro proceso de aplicación, o incluso del mismo proceso de aplicación, de actualizar una fila de la tabla base de la sentencia SELECT para que una fila del cursor ya no tenga ninguna fila correspondiente en la tabla base. Aquí, puede existir un “hueco de actualización”. Véase “hueco”. Véase también “hueco de supresión”.

**hueco de supresión.** Es posible que otro proceso de aplicación o incluso el mismo proceso de aplicación suprima una fila de tabla base de la sentencia SELECT de modo que una fila del cursor ya no tenga ninguna fila correspondiente en la tabla base. Aquí, existe efectivamente un “hueco de supresión” y ya no se puede acceder a dicha fila mediante el cursor. Véase “hueco”. Véase también “hueco de actualización”.

**Hypertext Markup Language (HTML).** Método estándar para presentar datos a los usuarios de Web. Las páginas Web están diseñadas utilizando códigos HTML en el texto. Estos códigos definen el diseño de la página, los gráficos y los enlaces de hipertexto en el documento y en otros documentos de Internet.

**ICAPI (Internet Connection API).** API de conexión a Internet.

**ICF (integrated catalog facility).** En un entorno OS/390, servicio de catálogo integrado.

**IDCAMS.** En un entorno OS/390, programa de IBM que se utiliza para procesar mandatos de los servicios del método de acceso. Se puede invocar como trabajo o paso de trabajo, desde un terminal TSO o desde un programa de aplicación del usuario.

**ID de aplicación.** Es una serie exclusiva que se genera cuando la aplicación se conecta a la base de datos o cuando DB2 Connect recibe una petición de conexión a una base de datos de Arquitectura de bases de datos relacionales distribuidas (“Arquitectura de Bases de Datos Relacionales Distribuidas” en la página 1531). Se genera un identificador cuando la aplicación se conecta a la base de datos. Este ID es conocido en el cliente y en el servidor y se puede utilizar para correlacionar las dos partes de la aplicación.

**ID de autorización.** (1) En una sentencia, serie de caracteres que designa un conjunto de privilegios. El gestor de bases de datos utiliza este ID para comprobar autorizaciones y como calificador implícito de los nombres de objetos, tales como tablas, vistas e índices. (2) Serie que se puede verificar para la conexión a DB2 Universal Database y a la que se aplica un conjunto de privilegios. Un identificador de autorización puede representar un individuo, un grupo de organización o una función, pero DB2 Universal Database no determina esta representación.

**ID de autorización de SQL (ID de SQL).** En DB2 Universal Database para OS/390, identificador de autorización que se utiliza para comprobar sentencias de SQL dinámicas en algunas situaciones.

**ID de autorización primario.** Identificador de autorización utilizado para identificar el proceso de aplicación en DB2 Universal Database para OS/390.

**ID de autorización secundario.** En DB2 Universal Database para OS/390, identificador de autorización que se asocia con un ID de autorización primario mediante una rutina de salida de autorización.

**ID de conexión.** En DB2 Universal Database para OS/390, identificador suministrado por el recurso de conexión y que está asociado a una conexión específica de espacio de direcciones.

**ID de correlación.** En DB2 Universal Database para OS/390, identificador que está asociado a una hebra determinada. En TSO, es un identificador de autorización o el nombre de trabajo.

**ID de RED.** Identificador de la red SNA donde reside la LU remota. Este ID de red es una serie de uno a ocho dígitos que sigue el convenio de denominación de SNA.

**ID de SQL.** Véase “ID de autorización de SQL”.

**ID de SQL actual.** Identificación que, en un momento determinado, tiene los privilegios que se ejercen cuando se ejecutan determinadas sentencias de SQL dinámicas. El ID de SQL actual puede ser un ID de autorización primaria o un ID de autorización secundaria.

**identificación.** Petición que un programa de servicio de conexión (de un espacio de direcciones que está separado de DB2 Universal Database para OS/390) emite mediante la interfaz del subsistema MVS para informar de su existencia a DB2 Universal Database para OS/390 e iniciar el proceso de conexión a DB2.



## Glosario

**identificador de componentes del recurso de instrumentación.** En DB2 Universal Database para OS/390, valor que designa e identifica un registro de rastreo de un suceso rastreado. Como parámetro de los mandatos START TRACE y MODIFY TRACE, especifica que debe rastrearse el suceso correspondiente.

**identificador de fila (ROWID).** Valor que identifica de forma exclusiva una fila. Este valor se almacena con la fila y no cambia.

**identificador de juego de caracteres codificado (CCSID).** Número que comprende un identificador de esquema de codificación, identificadores de juegos de caracteres, identificadores de páginas de códigos e información adicional que identifica de forma exclusiva la representación codificada de caracteres gráficos.

**identificador delimitado.** Secuencia de caracteres entre comillas dobles ("). La secuencia debe estar formada por una letra seguida de cero o más caracteres, que pueden ser una letra, un dígito o el carácter de subrayado. Véase también "identificador ordinario".

**identificador de red (NID).** En un entorno OS/390, identificador de red asignado por IMS o CICS o, si el tipo de conexión es RRSAF, identificador de unidad de recuperación (URID) de OS/390 RRS.

**identificador de registro (RID).** Número de página de 3 bytes seguido de un número de ranura de un byte que DB2 utiliza internamente para identificar de forma exclusiva un registro de una tabla. El RID contiene información suficiente para direccionar la página en la que está almacenado el registro. Véase también "identificador de fila".

**identificador de sistema principal.** Nombre declarado en el programa de sistema principal.

**identificador de tabla.** Calificador de nombres de columna que designa una determinada tabla objeto.

**identificador de unidad lógica de trabajo (LUWID).** En un entorno OS/390, nombre que identifica de forma exclusiva una hebra dentro de una red. Este nombre consta de un nombre de red completamente calificado de la LU, un número de instancia de LUW y un número de secuencia de LUW.

**identificador ordinario.** (1) En SQL, letra individual o seguida de caracteres, que pueden ser una letra (a-z y A-Z), un símbolo, un número o el carácter de subrayado, utilizados para formar un nombre. (2) En DB2 Universal Database para OS/390, letra *mayúscula* seguida de cero o más caracteres, cada uno de los cuales es una letra *mayúscula*, un número o el carácter de subrayado. Un identificador común no debe ser una palabra reservada. Véase también "identificador delimitado".

**Identificador R.** Véase "identificador de registro".

**IFCID (instrumentation facility component identifier).** En DB2 Universal Database para OS/390, identificador de componentes del recurso de instrumentación.

**IFI (instrumentation facility interface).** En DB2 Universal Database para OS/390, interfaz del recurso de instrumentación.

**IFP (IMS Fast Path).** En un entorno OS/390, vía rápida de IMS.

**ILU.** Véase "unidad lógica independiente" en la página 1621.



**imagen anterior.** En el proceso de duplicación de datos de DB2, contenido de un elemento de una tabla fuente antes de una renovación, tal como está registrado en una tabla de datos de cambios, en un archivo de anotaciones de base de datos o en un diario. Véase también “imagen-posterior”.

**imagen-posterior.** En el proceso de duplicación de datos de DB2, contenido actualizado de un elemento de una tabla fuente que se registra en una tabla de datos de cambios, en un archivo de anotaciones de base de datos o en un diario. Véase también “imagen anterior”.

**implantador de función.** En DB2 Universal Database para OS/390, identificador de autorización del propietario del programa de función y del paquete de función.

**importar.** Copiar datos de un archivo externo, utilizando formatos tales como PC/IXF, DEL, WSF o ASC, a las tablas del gestor de bases de datos. Véase también “exportar” en la página 1562.

**importar metadatos.** Proceso de trasladar metadatos al Centro de depósito de datos, ya sea de forma dinámica (desde la interfaz de usuario) o por lotes.

**IMS (Information Management System).** Sistema de gestión de información.

**independiente.** En DB2 Universal Database para OS/390, objeto (fila, tabla o espacio de tablas) que no es un objeto padre ni un objeto dependiente de otro.

**indicación de fecha y hora.** Valor de siete partes que consta de una fecha y una hora expresadas en años, meses, días, horas, minutos, segundos y microsegundos.

**indicador nulo.** Columna (por posición de byte) de un archivo ASCII no delimitado que contiene el “distintivo de indicador nulo” en la página 1555 para los datos que se están cargando en una columna de tabla. El valor nulo puede ser cualquier entero positivo válido.

**índice.** Conjunto de punteros que están ordenados lógicamente según los valores de una clave. Los índices proporcionan un acceso rápido a los datos y pueden asegurar la unicidad en las filas de la tabla. Cuando se solicita un índice, el gestor de bases de datos crea la estructura y la mantiene automáticamente. El gestor de bases de datos utiliza el índice para mejorar el rendimiento y asegurar la exclusividad.

**índice auxiliar.** En DB2 Universal Database para OS/390, índice de una tabla auxiliar en la que cada entrada de índice hace referencia a un LOB. Véase también “tabla auxiliar” en la página 1613.

**índice con cluster.** Índice cuya secuencia de valores de clave corresponde en gran medida a la secuencia de filas almacenadas en una tabla. El grado con que se da esta correspondencia se mide mediante estadísticas que utiliza el optimizador. Sinónimo de “índice de particionamiento”.

**índice de mapa de particionamiento.** Número asignado a una partición de hash o partición de rango.

**índice de particionamiento.** Índice que determina cuántas filas se ordenan físicamente en un espacio de tablas particionado. Sinónimo de “índice con cluster”.

**índice de unicidad.** Índice que asegura que no se almacenen valores de clave idénticos en una tabla.

**índice primario.** En DB2 Universal Database para OS/390, índice que asegura la unicidad de una clave primaria.

## Glosario

**índices de tipo 1.** Índices que se han creado mediante un release de DB2 anterior a DB2 para MVS/ESA Versión 4 o que están especificados como índices de tipo 1 en la Versión 4. En la fecha de DB2 Universal Database para OS/390 Versión 7, los índices de tipo 1 ya no se soportan. Véase también “índices de tipo 2”.

**índices de tipo 2.** Índices creados en un release de DB2 posterior a DB2 para OS/390 Versión 6 o que están especificados como índices de tipo 2 en la Versión 4 o Versión 6. Véase también “índices de tipo 1”.

**índice sin particionamiento.** En DB2 Universal Database para OS/390, cualquier índice que no sea un índice de particionamiento. Por ejemplo, si se define un índice sin particionamiento y un índice de particionamiento en la misma tabla, se pierden algunas de las ventajas de la independencia de nivel de partición para las operaciones de programa de utilidad, porque el acceso a un índice sin particionamiento debe ser secuencial.

**inhabilitar.** Restaurar una base de datos, una tabla de texto o una columna de texto a la condición que tenía antes de habilitarse para Text Extender eliminando los elementos creados durante el proceso de habilitación.

**inicialización del archivo de anotaciones.** Primera fase del proceso de reinicio durante la cual DB2 Universal Database para OS/390 intenta localizar el fin actual del archivo de anotaciones.

**inicio de sesión.** Petición que realiza un recurso de conexión en nombre de un proceso de aplicación CICS o IMS individual para permitir que DB2 Universal Database para OS/390 verifique si está autorizado a utilizar recursos de DB2.

**inmovilizar.** Finalizar un proceso permitiendo que las operaciones se completen normalmente, mientras se rechazan las peticiones de trabajo nuevas.

**inscripción.** Véase “fuente de duplicación” en la página 1563.

**instalación distribuida.** Proceso mediante el cual se pueden instalar productos DB2 utilizando software de gestión de sistemas, por ejemplo Microsoft Systems Management Server (SMS) en Windows NT o Windows 2000, o simplemente con una unidad de CD-ROM compartida o una unidad disco duro de red compartida utilizando archivos de respuestas. También se conoce como instalación silenciosa o instalación desatendida.

**instancia.** (1) Véase “instancia de gestor de bases de datos”. (2) Entorno lógico del servidor de expansores de DB2. Puede haber varias instancias del servidor de expansores de DB2 en la misma estación de trabajo, pero una sola instancia para cada instancia de DB2.

**instancia de gestor de bases de datos.** (1) Entorno lógico del gestor de bases de datos similar a una imagen del entorno real del gestor de bases de datos. Es posible tener varias instancias del producto gestor de bases de datos en la misma estación de trabajo. Estas instancias se utilizan para separar el entorno de desarrollo del entorno de producción, ajustar el gestor de bases de datos a un entorno determinado y proteger información sensible. (2) Código de DB2 que gestiona datos. Una instancia tiene sus propias bases de datos (a las que otras instancias no pueden acceder) y todas las particiones de base de datos comparten los mismos directorios del sistema. También tiene seguridad independiente de otras instancias de la misma máquina (sistema).

**instantánea.** Una instantánea es un registro del estado actual del entorno de base de datos. Véase también “instantánea de rendimiento” en la página 1573 y 1562.

**instantánea de explicación.** (1) Conjunto de información que se comprime cuando se explica una sentencia de SQL. (2) Captura de la representación interna actual de una consulta de SQL y de la información relacionada. Esta información es necesaria para la herramienta Visual Explain.

**instantánea de rendimiento.** Datos de rendimiento para un conjunto de objetos de la base de datos que se recuperan desde el gestor de bases de datos en un momento determinado.

**integridad de comprobación.** Condición que existe cuando cada fila de una tabla se ajusta a las restricciones de comprobación de tabla que se han definido en dicha tabla. Para mantener la integridad de comprobación es necesario que DB2 imponga restricciones de comprobación de tabla en las operaciones que añaden o cambian datos.

**integridad de referencia.** Estado de una base de datos en el que son válidos todos los valores de todas las claves foráneas. Para mantener la integridad de referencia es necesario imponer las “restricciones de referencia” en la página 1604 en todas las operaciones que cambian los datos de una tabla en la que se definen dichas restricciones.

**Interactive System Productivity Facility (ISPF).** En un entorno OS/390, programa bajo licencia de IBM que proporciona servicios de diálogo interactivo. Estos paneles permiten al usuario realizar la mayoría de las tareas de DB2 de forma interactiva.

**interés de lectura/escritura inter-DB2.** En DB2 Universal Database para OS/390, propiedad de los datos de un espacio de tablas, índice o partición que ha sido abierto por más de un miembro de un grupo de compartimiento de datos y que se ha abierto para escribir por parte de uno de estos miembros como mínimo.

**interfaz administrativa del Centro de depósito de datos.** Interfaz de usuario existente con las funciones administrativas del Centro de depósito de datos. La interfaz sólo puede residir en el servidor del Centro de depósito de datos o en máquinas diferentes correspondientes a varios administradores.

**interfaz de archivos planos.** Conjunto de funciones incorporadas de Net.Data que permiten leer y grabar datos de archivos de texto corrientes.

**interfaz del recurso de instrumentación (instrumentation facility interface, IFI).** Interfaz de programación que permite a los programas obtener datos de rastreo en línea referentes a DB2 Universal Database para OS/390, someter mandatos de DB2 Universal Database para OS/390 y pasar datos a DB2 Universal Database para OS/390.

**interfaz de nivel de llamada (CLI).** API que se puede llamar para acceder a una base de datos que constituye una alternativa a una API de SQL incorporado. A diferencia del SQL incorporado, la CLI no necesita precompilarse o enlazarse a una base de datos, sino que, en lugar de ello, proporciona un conjunto estándar de funciones para procesar sentencias de SQL y servicios relacionados en la ejecución. Véase también “Interfaz de nivel de llamada de DB2”.

**Interfaz de nivel de llamada (CLI) de DB2.** La aplicación utiliza un conjunto estándar de funciones para procesar sentencias de SQL y servicios relacionados en la ejecución. No tiene que precompilarse o enlazarse.

**interfaz de programación de aplicaciones (API).** (1) Interfaz funcional proporcionada por el sistema operativo o por un programa bajo licencia adquirible por separado. La API permite que un programa de aplicación escrito en un lenguaje de alto nivel utilice datos o funciones determinados del sistema

## Glosario

operativo o de los programas bajo licencia. (2) En DB2, las API permiten realizar la mayoría de las funciones administrativas desde dentro de un programa de aplicación.

**Interfaz remota de aplicaciones de bases de datos (DARI).** Término obsoleto para “procedimiento almacenado” en la página 1594.

**internacionalización.** En el entorno OS/390, soporte para un esquema de codificación que es capaz de representar los elementos de código de los caracteres de muchas geografías y muchos idiomas diferentes. Para soportar todas las geografías, el estándar Unicode necesita más de 1 byte para representar un solo carácter. Véase también “Unicode” en la página 1621.

**internal resource lock manager (IRLM).** Componente de DB2 Universal Database para OS/390. Cada subsistema DB2 debe tener su propia instancia del IRLM (gestor de bloqueos de recursos interno). El IRLM funciona con DB2 para controlar el acceso a los datos. DB2 solicita bloqueos al IRLM para asegurar la integridad de los datos cuando las aplicaciones, los programas de utilidad y los mandatos intentan acceder a los mismos datos.

**Internet Protocol (Protocolo Internet - IP).** En un entorno Internet, protocolo utilizado para encaminar datos desde su fuente hasta su destino. Véase también “Transmission Control Protocol/Internet Protocol” en la página 1620.

**Internetwork Packet Exchange (IPX).** Protocolo de datagramas sin conexión que se utiliza en un entorno de LAN NetWare para transferir datos a un nodo remoto. IPX intenta siempre enviar los paquetes de datos, pero no asegura la entrega fiable de los datos.

**intervalo de control.** En VSAM, espacio fijo de almacenamiento de acceso directo en el cual VSAM almacena registros y crea espacio libre distribuido. En un archivo ordenado por clave, un intervalo de control es el conjunto de registros a los que apunta una entrada del registro de índice. El intervalo de control es la unidad de información que se intercambia entre VSAM y el almacenamiento de acceso directo. Un intervalo de control incluye siempre un número entero de registros físicos.

**intervalo libre de vuelta atrás.** Conjunto de registros de anotación cronológica que no están compensados si la transacción termina anormalmente. Véase “vuelta atrás” en la página 1626.

**inversión de bytes.** Técnica en la cual los datos numéricos se almacenan con el byte menos significativo ocupando el primer lugar.

**invocación de función.** Uso de una función junto con los valores argumento que se pasan al cuerpo de la función. La función se invoca por su nombre.

**IP.** Véase “Internet Protocol”.

**IPX.** Véase “Internetwork Packet Exchange”.

**IRLM (internal resource lock manager).** Véase “internal resource lock manager”.

**ISAPI.** Internet Server API (API de servidor Internet) de Microsoft.

**ISPF.** Véase “Interactive System Productivity Facility” en la página 1573.

**ISPF/PDF.** En un entorno OS/390, Interactive System Productivity Facility/Program Development Facility.

## J

**JCL.** Véase “lenguaje de control de trabajos” en la página 1576.

**jerarquía de objetos de base de datos.** Organización de objetos de la base de datos en relaciones padre/hijo. Por ejemplo, las bases de datos son hijos del padre de la instancia de base de datos.

**JES.** Véase “Job Entry Subsystem”.

**Job Entry Subsystem (JES).** Programa bajo licencia de IBM que recibe trabajos en el seno del sistema y procesa los datos de salida producidos por los trabajos.

**juego de caracteres .** Juego definido de caracteres. Por ejemplo, 26 letras no acentuadas de la A a la Z.

**juego de caracteres codificados.** Conjunto de normas no ambiguas que definen un juego de caracteres y las relaciones biunívocas entre los caracteres del juego y sus representaciones codificadas.

**juego de caracteres de doble byte (DBCS).** Juego de caracteres en el que cada carácter está representado por dos bytes. Estos juegos de caracteres los utilizan normalmente los idiomas nacionales como, por ejemplo, el japonés y el chino, que tienen más símbolos de los que pueden representarse mediante un solo byte. Véase también “juego de caracteres de un solo byte” y “juego de caracteres de múltiples bytes”.

**juego de caracteres de múltiples bytes (MBCS).** Juego de caracteres en el que cada carácter está representado por 2 o más bytes. Los juegos de caracteres que sólo utilizan dos bytes se conocen más comúnmente como juegos de caracteres de doble byte. Véase también “ASCII” en la página 1531, “juego de caracteres de un solo byte”, “EBCDIC” en la página 1556 y “Unicode” en la página 1621.

**juego de caracteres de un solo byte (SBCS).** Juego de caracteres en el que cada carácter se representa mediante un código de un solo byte. Véase también “juego de caracteres de doble byte” y “juego de caracteres de múltiples bytes”.

**juego de caracteres invariable.** (1) Juego de caracteres, por ejemplo el juego de caracteres sintácticos, cuyas asignaciones de elementos de código no cambian de una página de códigos a otra. (2) Conjunto mínimo de caracteres que está disponible como parte de todos los juegos de caracteres. Véase también “juego de caracteres sintácticos”.

**juego de caracteres sintácticos.** Conjunto de 81 caracteres gráficos registrados en el registro de IBM como juego de caracteres 00640. Originalmente este juego de caracteres se recomendaba para los usuarios de lenguajes de programación para su uso con fines sintácticos, a fin de maximizar la portabilidad y posibilidad de intercambio entre sistemas y países. Está incluido en la mayoría de juegos de caracteres registrados principales, con unas pocas excepciones. Véase también “juego de caracteres invariable”.

**juego de códigos.** Término de la ISO (International Organization for Standardization - Organización internacional para la estandarización) para página de códigos. Véase “página de códigos” en la página 1587.

## Glosario

### K

**Kerberos** . Protocolo de autenticación de red que está diseñado para proporcionar autenticación potente para aplicaciones cliente/servidor utilizando criptografía de clave secreta. Véase también “certificado de Kerberos” en la página 1538.

**KSDS**. Véase “conjunto de datos ordenado por clave” en la página 1543.

### L

**Language Environment**. Módulo que proporciona acceso desde una macro de Net.Data a una fuente de datos externa, tal como DB2, o a un lenguaje de programación, tal como Perl.

**LCID (log control interval definition)**. En un entorno OS/390, definición del intervalo de control del registro cronológico.

**LDS**. Véase “conjunto de datos lineal” en la página 1543.

**lectura diseminada**. Método del gestor de bases de datos para leer páginas de datos contiguas de disco en partes discontiguas de memoria. Véase también “E/S basada en bloque” en la página 1558.

**lectura no confirmada (UR)**. Nivel de aislamiento que permite a una aplicación acceder a cambios no confirmados de otras transacciones. La aplicación no bloquea otras aplicaciones fuera de la fila que está leyendo, a no ser que la otra aplicación intente eliminar o modificar la tabla. Véase también “lectura repetible”, “estabilidad del cursor” en la página 1560 y “estabilidad de lectura” en la página 1560.

**lectura repetible (RR)**. Nivel de aislamiento que bloquea todas las filas de una aplicación a las que se hace referencia en una transacción. Cuando un programa utiliza la protección de lectura repetible, las filas a las que hace referencia el programa no pueden ser modificadas por otros programas hasta que el programa finaliza la transacción actual. Véase también “estabilidad de lectura” en la página 1560, “lectura no confirmada (UR)” y “estabilidad del cursor” en la página 1560.

**Lenguaje de Consulta Estructurada (Structured Query Language - SQL)**. Lenguaje estandarizado para definir y manipular datos de una base de datos relacional.

**lenguaje de control de trabajos (JCL)**. Lenguaje de control que se utiliza para identificar un trabajo ante un sistema operativo y describir los requisitos del trabajo.

**lenguaje de definición de datos (data definition language - DDL)**. Lenguaje para describir datos y sus relaciones en una base de datos.

**lenguaje de descripción de datos**. Sinónimo de “lenguaje de definición de datos”.

**lenguaje de manipulación de datos (DML)**. Subconjunto de sentencias de SQL utilizadas para manipular datos. La mayoría de las aplicaciones utilizan principalmente sentencias de SQL DML, soportadas por el programa DB2 Connect. Las sentencias SELECT, INSERT, UPDATE y DELETE son similares en los productos de bases de datos relacionales de IBM.

**lenguaje de marcación extensible (extensible markup language - XML)**. Lenguaje de códigos basado en texto utilizado para el proceso de documentos y para publicar información en la Web.

**lenguaje principal.** Cualquier lenguaje de programación donde el usuario puede intercalar sentencias de SQL.

**límite de sesiones.** En SNA, número máximo de sesiones activas simultáneas de unidad lógica a unidad lógica (LU-LU) a las que puede dar soporte una unidad lógica determinada.

**lista.** Tipo de objeto, que los programas de utilidad de DB2 pueden procesar, que identifica múltiples espacios de tablas y/o múltiples espacios de índice. Una lista se define con la sentencia de control de programa de utilidad LISTDEF.

**lista de mandatos.** Lenguaje que DB2 Universal Database para OS/390 utiliza para realizar tareas de TSO.

**lista de páginas lógicas (LPL).** En DB2 Universal Database para OS/390, lista de páginas erróneas a las que no pueden hacer referencia las aplicaciones hasta que se recuperen las páginas. La página tiene un error lógico, aunque puede que el soporte de almacenamiento real (recurso de acoplamiento o DASD) no contenga ningún error. Generalmente, se ha perdido una conexión con el soporte de almacenamiento.

**lista de paquetes.** En DB2 Universal Database para OS/390, lista ordenada de nombres de paquetes que se pueden utilizar para ampliar un plan de aplicación.

**lista enumerada.** En el entorno OS/390, conjunto de objetos DB2 que se definen con una sentencia de control del programa de utilidad LISTDEF en la que se utilizan caracteres que coinciden con un patrón (\*, % o ?).

**LISTCAT de IDCAMS.** En un entorno OS/390, recurso para obtener información que está contenida en el catálogo de los servicios del método de acceso.

**Live Connection.** Componente de Net.Data que consta de un Gestor de conexiones y múltiples clientes. Live Connection gestiona la reutilización de las conexiones de base de datos y de máquina virtual Java.

**llamada de IFI.** En DB2 Universal Database para OS/390, invocación de la interfaz del recurso de instrumentación (IFI) por medio de una de sus funciones definidas.

**LOB.** Véase “objeto grande” en la página 1586.

**local.** Forma de hacer referencia a cualquier objeto mantenido por el subsistema local. Por ejemplo, en DB2 Universal Database para OS/390, un tabla local es una tabla mantenida por el subsistema DB2 local. Véase también “remoto” en la página 1602.

**localizador.** Véase “localizador de LOB”.

**localizador de conjunto resultante.** Valor de 4 bytes que DB2 Universal Database para OS/390 utiliza para identificar exclusivamente al conjunto resultante de una consulta devuelto por un procedimiento almacenado.

**localizador de LOB.** Mecanismo que permite que un programa de aplicación maneje un valor de objeto grande en el sistema de base de datos. Un localizador LOB es un valor de símbolo simple que representa un solo valor de LOB. Un programa de aplicación recupera un localizador de LOB en una variable del lenguaje principal y entonces puede aplicar las funciones de SQL al valor de LOB asociado utilizando el localizador.



## Glosario

**localizador de tabla.** En DB2 Universal Database para OS/390, mecanismo que permite acceder a tablas de transición de activación de la cláusula FROM de sentencias SELECT, la subselección de sentencias INSERT o desde funciones definidas por el usuario. Un localizador de tabla es un valor entero de palabra completa que representa una tabla de transición.

**localizador uniforme de recursos (uniform resource locator - URL).** Dirección Web, que ofrece un modo de nombrar y localizar elementos específicos en la Web.

**longitud de registro.** Suma de la longitud de todas las columnas de una tabla, que es la longitud de los datos tal como están físicamente almacenados en la base de datos. Los registros pueden tener una longitud fija o variable, en función de cómo se definan las columnas. Si todas las columnas son de longitud fija, el registro es de longitud fija. Si una o varias columnas tienen una longitud variable, el registro tiene una longitud de columna variable.

**LPL.** Véase “lista de páginas lógicas” en la página 1577.

**LRECP.** Véase “pendiente de recuperación lógica” en la página 1591.

**LRH (log record header).** Véase “cabecera de fila lógica” en la página 1535.

**LRSN (log record sequence number).** Véase “número de secuencia de registro de anotaciones” en la página 1586.

**LU 6.2.** Véase “unidad lógica 6.2” en la página 1621.

**LU.** Véase “unidad lógica” en la página 1621.

**LUW.** Véase “unidad lógica de trabajo” en la página 1621.

**LUID.** Véase “identificador de unidad lógica de trabajo” en la página 1570.

## M

**mandato.** (1) Modo de ejecutar funciones de administración de base de datos para acceder al gestor de bases de datos y mantenerlo. (2) En DB2 Universal Database para OS/390, mandato de operador o submandato de DSN. Un mandato es diferente de una sentencia de SQL. Véase “mandato de DB2”.

**mandato de DB2.** Instrucción al sistema operativo para acceder y mantener el gestor de bases de datos. Por ejemplo, los mandatos de DB2 permiten a un usuario iniciar o detener una base de datos y visualizar información sobre los usuarios actuales y el estado de las bases de datos.

**mapa de extensiones.** Estructura de metadatos almacenada en un espacio de tablas que registra la asignación de extensiones a cada objeto del espacio de tablas.

**mapa de particionamiento.** Vector de números de partición que correlaciona un índice de mapa de particionamiento con particiones de base de datos del grupo de particiones de base de datos.

**máquina de sistema principal.** (1) En una red de sistemas, sistema que proporciona servicios tales como proceso, acceso a bases de datos y funciones de control de la red. (2) Sistema primario o de control en una instalación que consta de varios sistemas.

**marcador.** Indicador del valor actual de un elemento. Véase también “contador” en la página 1544.



**marcador de parámetro.** Signo de interrogación (?) que aparece en una serie de una sentencia de SQL dinámica. El signo de interrogación puede aparecer donde podría haber una variable de lenguaje principal si la sentencia fuera una sentencia de SQL estático.

**marcador de parámetro con tipo.** Marcador de parámetro que se especifica junto con su tipo de datos de destino. Tiene el formato general: *CAST (? AS tipo-datos)*.

**marcador de parámetros sin tipo.** Marcador de parámetros que se especifica sin su tipo de datos de destino. Adopta la forma de un signo de interrogación individual.

**materializar.** (1) En DB2 Universal Database para OS/390, proceso de colocación de filas de una vista o expresión de tabla anidada en un archivo de trabajo para su proceso adicional por una consulta. (2) Colocación de un valor de LOB en almacenamiento contiguo. Debido a que los valores LOB pueden ser muy grandes, DB2 Universal Database para OS/390 evita materializar datos LOB hasta que no sea absolutamente necesario.

**MBCS.** Véase “juego de caracteres de múltiples bytes” en la página 1575.

**mensajes de notificación de administración.** Errores, avisos y mensajes informativos escritos por DB2, los programas Capture y Apply y las aplicaciones de usuario en un archivo de notificación o una anotación cronológica de sucesos.

**menú.** En DB2 Universal Database para OS/390, lista visualizada de funciones que puede seleccionar el operador. A veces se denomina *panel de menú*.

**metadatos.** Datos que describen las características de datos almacenados; datos descriptivos. Por ejemplo, los metadatos de una tabla de base de datos pueden contener el nombre de la tabla, el nombre de la base de datos donde reside la tabla, los nombres de las columnas de la tabla y las descripciones de columnas, ya sea en términos técnicos o comerciales. Los catálogos de base de datos y los catálogos de información contienen metadatos.

**metadatos comerciales.** Datos que describen recursos de información en términos comerciales. Los metadatos comerciales se almacenan en el catálogo de información y los usuarios acceden a ellos para buscar e interpretar la información que necesitan. Por ejemplo, los metadatos comerciales para un programa contendrán una descripción de lo que hace el programa y de las tablas que utiliza. Véase también “metadatos técnicos”.

**metadatos de control.** En Centro de depósito de datos, información sobre cambios realizados en el depósito de datos, tales como la fecha y la hora en que un paso de proceso actualiza una tabla.

**metadatos de definición.** En Centro de depósito de datos, información sobre el formato del depósito de datos (el esquema), las fuentes de los datos y las transformaciones aplicadas al cargar los datos.

**metadatos técnicos.** En Centro de depósito de datos, datos que describen los aspectos técnicos de los datos, tales como el tipo y la longitud de su base de datos. Los metadatos técnicos incluyen información sobre el origen de los datos y las reglas utilizadas para extraer, depurar y transformar los datos. Gran parte de los metadatos del Centro de depósito de datos son técnicos. Véase también “metadatos comerciales”.

**método de acceso secuencial básico (basic sequential access method - BSAM).** Método de acceso que DB2 Universal Database para OS/390 utiliza para almacenar o recuperar bloques de datos en una

## Glosario

secuencia continua, utilizando un dispositivo de acceso secuencial o de acceso directo. Véase también “método de acceso secuencial en cola”.

**método de acceso secuencial en cola (QSAM).** Versión ampliada del “método de acceso secuencial básico” en la página 1579 (BSAM). Cuando se utiliza este método, se forma una cola de bloques de datos de entrada que están a la espera de ser procesados, o una cola de bloques de datos de salida que están a la espera de ser transferidos a almacenamiento auxiliar o a un dispositivo de salida.

**métrica de rendimiento.** Conjunto de todas las variables de rendimiento que pertenecen al mismo objeto de base de datos.

**miembro.** (1) En DB2 Universal Database, miembro de conjunto de suscripción. (2) En el Kit de iniciación de OLAP, método para hacer referencia a datos mediante tres o más dimensiones. Un valor de datos individual de una tabla factual es la intersección de un miembro de cada dimensión.

**miembro de compartimiento de datos.** Subsistema de DB2 Universal Database para OS/390 que está asignado a un grupo de compartimiento de datos por los servicios de XCF.

**miembro de conjunto de suscripción.** En el proceso de duplicación de datos de DB2, miembro de un conjunto de suscripción. Existe un miembro por cada par fuente-destino. Cada miembro define la estructura de la tabla destino y qué filas y columnas se duplicarán de la tabla fuente.

**migración.** (1) Proceso de trasladar datos desde un sistema a otro sin convertir los datos. (2) Instalación de una versión o un release nuevo de un programa para sustituir una versión o release anterior.

**minería de datos.** Proceso de recopilación de información de gestión crítica de un depósito de datos, de correlación de la misma y de revelación de las asociaciones, los patrones y las tendencias.

**modalidad.** En Centro de depósito de datos, fase de desarrollo de un paso de proceso, tal como desarrollo, prueba o producción.

**modalidad de bloqueo.** Representación del tipo de acceso que pueden tener programas en ejecución simultánea para un recurso bloqueado por DB2 Universal Database para OS/390.

**modalidad de duplicación móvil.** Modalidad de duplicación en la que los programas Capture y Apply operan cuando es necesario en lugar de hacerlo de forma autónoma y continua. Esta modalidad se invoca desde el cliente móvil y permite reproducir datos cuando el cliente móvil está disponible para conectarse al servidor fuente o destino.

**MODEENT.** En un entorno OS/390 macroinstrucción de VTAM que asocia un nombre de modalidad de conexión con un conjunto de parámetros que representan protocolos de sesión. Un conjunto de macroinstrucciones MODEENT define una tabla de modalidad de conexión.

**modelo de datos lógico.** Proceso de documentación de los requisitos de información comercial completa en un formato preciso y coherente. El modelo de datos es el primer paso del diseño de una base de datos.

**modelo de documento.** Definición de la estructura de un documento basada en las secciones que contiene el documento. Text Extender utiliza un modelo de documento al indexar.

**módulo de carga.** Unidad de programa que se puede cargar en almacenamiento principal para su ejecución. Es la salida de un editor de enlaces.

**módulo de petición de base de datos (database request module - DBRM).** Miembro de archivo creado por el precompilador de DB2 Universal Database para OS/390 y que contiene información sobre sentencias de SQL. Los DBRM se utilizan en el proceso de enlace lógico.

**motor de base de datos.** Parte del gestor de bases de datos que proporciona las funciones base y los archivos de configuración necesarios para utilizar la base de datos.

**MPP (massively parallel processing).** Véase “programa de proceso de mensajes” en la página 1595 o “proceso masivamente paralelo” en la página 1595.

**MTO (master terminal operator).** En un entorno OS/390, operador de terminal maestro.

**multidimensional.** En el Kit de iniciación de OLAP, método para hacer referencia a datos mediante tres o más dimensiones. Un valor de datos individual de una tabla factual es la intersección de un miembro de cada dimensión. Véase también “dimensión comercial” en la página 1554 y “dimensión” en la página 1554.

**multitarea.** Modalidad de operación que permite la ejecución simultánea o entrelazada de dos o más tareas.

**MVS.** Multiple Virtual Storage es el sistema operativo principal utilizado en los sistemas principales de IBM. Este sistema operativo gestiona grandes cantidades de memoria y espacio en disco.

**MVS/ESA.** Multiple Virtual Storage/Enterprise Systems Architecture. Se ha redenominado y se conoce más comúnmente como OS/390.

## N

**NAU.** Véase “unidad direccionable de red” en la página 1621.

**navegador.** (1) Función de Text Extender que permite visualizar texto en un monitor de sistema. (2) Programa que permite a los usuarios examinar datos pero no cambiarlos.

**NDS.** Véase “Servicios de directorio de red” en la página 1607.

**Network Information Service (NIS/NIS+).** En AIX, registro central de contraseñas, nodos, etc que se puede utilizar con DB2 Administration Server (DAS) en la administración de nombres de usuarios y grupos.

**NID.** Véase “identificador de red” en la página 1570.

**NIS/NIS+.** Véase “Network Information Service”.

**nivel de aislamiento.** (1) Característica de seguridad que determina cómo se bloquean los datos para otros procesos mientras se está accediendo a ellos. Véase también REPEATABLE READ, READ STABILITY, CURSOR STABILITY y UNCOMMITTED READ. (2) Atributo que define el grado de aislamiento de un proceso de aplicación respecto a otros procesos de aplicación que se ejecutan simultáneamente.

**nivel de autorización.** Proporciona un método de agrupación de privilegios y operaciones de programa de utilidad y mantenimiento del gestor de bases de datos de nivel más alto. Véase “autorización de carga” en la página 1532 y “autorización del sistema” en la página 1532.

## Glosario

**nivel de sincronización.** En APPC, especificación que indica si los programas de transacciones correspondientes intercambian peticiones de confirmación y respuestas.

**NN.** Véase “nodo de red”.

**nodo.** (1) En el proceso de particionamiento de una base de datos, sinónimo de “partición de base de datos” en la página 1590. (2) En hardware, sistema uniprocador o multiprocador simétrico (SMP) que forma parte de un sistema de clusters o de un sistema de proceso masivo en paralelo (MPP). Por ejemplo, RS/6000 SP es un sistema MPP que consta de un número de nodos conectados mediante una red de alta velocidad. (3) En comunicaciones, punto final de un enlace de comunicaciones o punto de unión común a dos o más enlaces de una red. Los nodos pueden ser procesadores, controladores de comunicaciones, controladores de clusters, terminales o estaciones de trabajo. Los nodos pueden variar en su capacidad de direccionamiento y en otras características funcionales.

**nodo coordinador.** Véase “partición coordinadora” en la página 1590.

**nodo de base de datos.** Véase “partición de base de datos” en la página 1590.

**nodo de catálogo.** Véase “partición de catálogo” en la página 1590.

**nodo de donde proceden los datos.** En DB2 Universal Database para OS/390, nodo del árbol de puntos de sincronismo que es responsable, además de otros gestores de recuperación o recursos, de coordinar la ejecución de una confirmación en dos fases.

**nodo de red de entrada limitada (nodo LEN).** Nodo de tipo 2.1 que da soporte a protocolos LU independientes, pero no da soporte a sesiones de CP-CP. Puede ser un nodo periférico conectado a un nodo de límite en una red de subárea, un nodo final conectado a un nodo de red APPN en una red APPN o un nodo conectado a igual directamente conectado a otro nodo LEN o nodo final APPN.

**nodo de red intermedio.** En APPN, nodo que forma parte de una ruta entre una unidad lógica fuente y una unidad lógica de destino, pero que no contiene a ninguna de las dos ni actúa como servidor de red para ellas.

**nodo de red (NN).** En APPN, nodo de la red que proporciona servicios de directorios distribuidos, intercambios de bases de datos de topología con otros nodos de red APPN y servicios de direccionamiento y sesión. Véase también “Red avanzada de igual a igual” en la página 1601.

**nodo de sistema principal.** En SNA, nodo de subárea que contiene un punto de control de servicios del sistema (SSCP), por ejemplo un sistema IBM System/390 con MVS y VTAM.

**nodo final.** En APPN, nodo que soporta sesiones entre su punto de control local y el punto de control de un nodo de red adyacente.

**nodo LEN.** Véase “nodo de red de entrada limitada”.

**nodo lógico.** Partición en un procesador que tiene más de una partición asignada. Véase “nodo”.

**nodos adyacentes.** Dos nodos conectados entre sí como mínimo por una ruta que no conecta a ningún otro nodo.

**nombre calificado por la red.** Nombre por el que se conoce una LU en toda una red SNA interconectada. Un nombre calificado por la red consta de un nombre de red que identifica la subred

individual y un nombre de LU de red. Los nombres calificados por la red son exclusivos en toda una red interconectada. También se denomina *nombre de LU calificado por la red* o *nombre de LU totalmente calificado*.

**nombre comercial.** En el Centro de depósito de datos, nombre descriptivo que puede asociarse con un objeto que también tiene un nombre físico. Los tipos de objetos que pueden tener nombres comerciales son las tablas, los archivos, las columnas o los campos. El nombre comercial puede utilizarse al realizar una búsqueda. También se pasa las herramientas del usuario final mediante los recursos de intercambio de metadatos de depósito.

**nombre de aplicación.** Nombre de la aplicación que se ejecuta en el cliente que la identifica en el gestor de bases de datos o en DB2 Connect. Se pasa del cliente al servidor para establecer la conexión de base de datos.

**nombre de base de datos.** Nombre(s) de identificación que los usuarios proporcionan como parte del mandato CREATE DATABASE o de la interfaz de programación de aplicaciones. Estos nombres deben ser exclusivos en la ubicación en la que se han catalogado.

**nombre de base de datos relacional (relational database name - RDBNAM).** Identificador exclusivo para un nombre de base de datos relacional en una red. En DB2 Universal Database para OS/390, debe ser el valor de la columna LOCATION de la tabla SYSIBM.LOCATIONS de la base de datos de comunicaciones. En las publicaciones de DB2 Universal Database para OS/390, se hace referencia al nombre de otro RDBMS como valor de LOCATION o nombre de ubicación.

**nombre de correlación.** Identificador que designa una tabla o vista dentro de una sentencia de SQL individual. Se puede definir en cualquier cláusula FROM o en la primera cláusula de una sentencia UPDATE o DELETE.

**nombre de CP.** Véase “nombre de punto de control” en la página 1584.

**nombre de definición de datos (ddname).** En DB2 Universal Database para OS/390, nombre de una sentencia de definición de datos (DD) que corresponde a un bloque de control de datos que contiene el mismo nombre.

**nombre de destino simbólico.** Especifica el nombre de un asociado remoto. El nombre corresponde a una entrada de la tabla de información complementaria de Comunicaciones CPI que contiene la información necesaria (nombre de LU asociada, nombre de modalidad, nombre de TP asociado) para que el cliente establezca una conexión APPC con el servidor.

**nombre de dispositivo.** Nombre reservado por el sistema o un controlador de dispositivo que hace referencia a un dispositivo específico. Por ejemplo, el nombre de dispositivo de DOS para el puerto paralelo es LPT1.

**nombre de dominio.** Nombre que las aplicaciones TCP/IP utilizan para hacer referencia a un sistema principal TCP/IP dentro de una red TCP/IP. Un nombre de dominio consta de una secuencia de nombres separados por puntos. Por ejemplo, www.ibm.com.

**nombre de función específico.** (1) Nombre que identifica de forma exclusiva una función en el sistema. Muchos nombres específicos pueden tener el mismo nombre de función. (2) En DB2 Universal Database para OS/390, determinada función definida por el usuario que el gestor de bases de datos ya conoce por su nombre específico. Muchas funciones definidas por el usuario pueden tener el mismo nombre de función. Cuando en la base de datos se define una función definida por el usuario, a cada función se le

## Glosario

asigna un nombre específico que es exclusivo dentro del esquema de la función. El usuario puede proporcionar este nombre o utilizar el valor por omisión.

**nombre de función sobrecargado.** Nombre de función para que el que existen varias funciones dentro de un esquema o vía de función. Las funciones que están dentro del mismo esquema deben tener firmas diferentes.

**nombre de grupo.** En un entorno OS/390, identificador de XCF correspondiente a un grupo de compartimiento de datos.

**nombre del programa de transacciones.** En conversaciones de LU 6.2 de SNA, nombre del programa situado en la unidad lógica remota que será el otro participante en la conversación.

**nombre de LU.** En un entorno OS/390, nombre mediante el cual VTAM hace referencia a un nodo en una red. Véase también “nombre de ubicación” en la página 1585.

**nombre de LU completamente calificado.** Véase “nombre calificado por la red” en la página 1582.

**nombre de miembro.** Identificador de XCF para un subsistema determinado de DB2 Universal Database para OS/390 en un grupo de compartimiento de datos.

**nombre de modalidad.** (1) En APPC, nombre que utiliza el iniciador de una sesión para designar las características deseadas para la sesión, por ejemplo los límites de longitud de los mensajes, el punto de sincronismo, la clase de servicio dentro de la red de transporte y las características de retardo y direccionamiento de la sesión. (2) En un entorno OS/390 nombre de VTAM para el conjunto de características físicas y lógicas y atributos de una sesión.

**nombre de objeto de base de seguridad.** Serie de caracteres de 48 bytes que contiene el nombre de un objeto de base de seguridad del servidor de archivos NetWare. El campo de configuración del gestor de bases de datos, nombre de objeto, representa de forma exclusiva una instancia de servidor DB2 y se almacena como un objeto en la base de seguridad de un servidor de archivos NetWare.

**nombre de paquete.** Nombre de un objeto que se crea mediante el mandato BIND, PRECOMPILE o REBIND. El objeto es una versión enlazada de un módulo de petición de base de datos (DBRM). El nombre consta de un nombre de ubicación, un ID de colección, un ID de paquete y un ID de versión.

**nombre de plan.** En DB2 Universal Database para OS/390, nombre de un plan de aplicación.

**nombre de principal.** Nombre por el que se conoce un principal en los servicios de seguridad de DCE (Distributed Computing Environment).

**nombre de punto de control.** Nombre calificado de red de un punto de control que consta de un calificador de identificador de red que identifica la red a la que pertenece el nodo de punto de control. Véase “punto de control” en la página 1597.

**nombre de recurso genérico.** En un entorno OS/390, nombre que VTAM utiliza para representar varios programas de aplicación que proporcionan la misma función a fin de gestionar la distribución y el equilibrado de sesiones en un entorno Parallel Sysplex.

**nombre de red.** En SNA, nombre simbólico mediante el cual los usuarios finales hacen referencia a una unidad direccionable de red (NAU), una estación de enlace o un enlace.

**nombre de servicio.** Nombre que proporciona un método simbólico para especificar el número de puerto que se debe utilizar en un nodo remoto. Para identificar una aplicación, la conexión TCP/IP necesita la dirección del nodo remoto y el número de puerto que se debe utilizar en el nodo remoto.

**nombre de subsistema por omisión (DSN).** (1) Nombre del procesador de mandatos TSO de DB2 Universal Database para OS/390, obtenido de los módulos y las macros. (2) Especifica el nombre del subsistema DB2 que puede conectarse al servidor de control. El valor por omisión para el nombre de subsistema es DSN. Este parámetro deber ser el primero.

**nombre de tres partes.** Nombre completo de una tabla, vista o seudónimo. Consta de un nombre de ubicación, un identificador de autorización y un nombre de objeto, separados por puntos.

**nombre de ubicación.** Nombre mediante el cual DB2 Universal Database para OS/390 hace referencia a un subsistema DB2 determinado dentro de una red de subsistemas. Véase también “nombre de LU” en la página 1584.

**nombre entrecomillado.** Véase “identificador delimitado” en la página 1570.

**nombre expuesto.** Nombre de correlación, tabla o nombre de vista especificado en una cláusula FROM para la que no se ha especificado ningún nombre de correlación.

**nombre-parámetro.** Identificador largo que designa un parámetro al que se puede hacer referencia en un procedimiento o una función definida por el usuario.

**normalización.** En las bases de datos, proceso de reestructuración de un modelo de datos reduciendo sus relaciones a sus formas más simples. Es un paso clave en la tarea de creación de un diseño de base de datos relacional lógica. La normalización ayuda a evitar redundancias e incoherencias en los datos. Una entidad se normaliza si cumple un conjunto de restricciones para una forma normal determinada (primera forma normal, segunda forma normal, etc). Véase también “desnormalización” en la página 1552 y “grupo de repetición” en la página 1567.

**NRE (network recovery element).** En un entorno OS/390, elemento de recuperación de la red.

**NUL.** En el lenguaje de programación C, carácter individual que indica el final de la serie.

**NULLIF.** En DB2 Universal Database para OS/390, función escalar que evalúa dos expresiones pasadas y devuelve NULL si los argumentos son iguales o el valor del primer argumento si no lo son.

**nulo.** Valor que indica la ausencia de información.

**número de coma flotante de precisión doble.** En SQL, representación aproximada de un número real en forma de 64 bits.

**número de coma flotante de precisión simple.** Representación aproximada de 32 bits de un número real.

**número de planificación de aplicación de origen (OASN).** En un entorno OS/390 con IMS, número de 4 bytes que se asigna secuencialmente a cada planificación de IMS a partir del último arranque en frío de IMS. El OASN se utiliza como identificador para una unidad de trabajo. En un formato de 8 bytes, los 4 primeros bytes contienen el número de planificación y los 4 últimos bytes contienen el número de puntos de sincronismo de IMS (*puntos de confirmación*) durante la planificación actual. El OASN forma parte del NID para una conexión a IMS.



## Glosario

**número de secuencia del registro de anotaciones (LRSN).** Número que DB2 Universal Database para OS/390 genera y asocia a cada registro de anotaciones. El LRSN también se utiliza para crear versiones de páginas. Los LRSN generados por un grupo de compartimiento de datos de DB2 Universal Database para OS/390 forman una secuencia estrictamente creciente para cada archivo de anotaciones DB2 y una secuencia estrictamente creciente para cada página del grupo de compartimiento de datos.

## O

**OASN .** Véase “número de planificación de aplicación de origen” en la página 1585.

**OBID (data object identifier).** En DB2 Universal Database para OS/390, identificador de objeto de datos.

**objeto.** (1) Cualquier elemento que se pueda crear o manipular con SQL—por ejemplo, tablas, vistas, índices o paquetes. (2) En la programación o el diseño orientado a objetos, abstracción que consta de datos y operaciones asociadas con dichos datos. (3) En NetWare, entidad que está definida en la red y por tanto se le otorga acceso al servidor de archivos.

**objeto de base de datos.** (1) Asociación en una base de datos a cualquier elemento que se pueda supervisar. (2) Cualquier elemento que se pueda crear o manipular con SQL. Por ejemplo, tablas, vistas, índices, paquetes, desencadenantes o espacios de tablas.

**objeto de bloqueo.** Recurso controlado por un bloqueo de DB2 Universal Database para OS/390.

**objeto grande (LOB).** Secuencia de bytes con un tamaño en un rango de 0 bytes a 2 gigabytes menos 1 byte. Puede ser de uno de estos tres tipos: “gran objeto binario” en la página 1566 (binario), “gran objeto de caracteres” en la página 1566 (carácter de un solo byte o mixto) o “gran objeto de caracteres de doble byte” en la página 1566 (carácter de doble byte).

**obtener página (getpage).** Operación en la que DB2 Universal Database para OS/390 accede a una página de datos.

**ODBC.** Véase “Open Database Connectivity”.

**OLAP (online analytical processing).** Véase “proceso analítico en línea” en la página 1594.

**Opción de compartimiento de tiempo (Time-Sharing Option - TSO).** Este proceso es necesario para enlazar paquetes y planes de aplicaciones y para ejecutar varias funciones en línea que se proporcionan con DB2 Universal Database para OS/390. Mediante el uso del recurso de conexión TSO, puede acceder a DB2 ejecutando en primer plano o en proceso por lotes.

**opción de control de vista.** En DB2 Universal Database para OS/390, opción que especifica si cada fila que se inserta o actualiza mediante una vista debe cumplir con la definición de esa vista. Se puede especificar una opción de control de vista en las cláusulas WITH CASCADED CHECK OPTION, WITH CHECK OPTION o WITH LOCAL CHECK OPTION de la sentencia CREATE VIEW.

**Open Database Connectivity (ODBC).** API que permite acceder a sistemas de gestión de bases de datos utilizando SQL invocable, el cual no necesita la utilización de un preprocesador de SQL. La arquitectura ODBC permite a los usuarios añadir módulos, denominados *controladores de bases de datos*, que durante la ejecución enlazan la aplicación con sistemas seleccionados de gestión de bases de datos. No es necesario enlazar directamente las aplicaciones con los módulos de todos los sistemas de gestión de bases de datos soportados.



**operación de SQL desencadenante.** Operación de SQL que hace que se active un desencadenante cuando se realiza en la tabla sujeto.

**operador.** Acción que debe realizarse en los datos o en la salida de una tabla o un índice, cuando se ejecuta el plan de acceso para una sentencia de SQL.

**operador de comparación.** Los operadores de comparación son  $\neq$  (no menor que),  $\leq$  (menor que o igual a),  $\neq$  (no igual a),  $=$  (igual a),  $\geq$  (mayor que o igual a),  $>$  (mayor que) y  $\neq$  (no mayor que). Véase también “operador de infijo”.

**operador de conjunto.** Los operadores UNION, EXCEPT e INTERSECT de SQL correspondientes a los operadores relacionales de unión, diferencia e intersección. Un operador de conjunto obtiene una tabla resultante combinando otras dos tablas resultantes.

**operador de infijo.** Operador utilizado en expresiones de comparación. Véase también “operador de comparación”.

**operador lógico.** Palabra clave que especifica cómo deben evaluarse varias condiciones de búsqueda (AND, OR) o si se debe invertir el sentido lógico de una condición de búsqueda (NOT).

**operando.** Entidad en la que se efectúa una operación.

**optimizador.** Componente del compilador de SQL que elige un plan de acceso para una sentencia de lenguaje de manejo de datos modelando el coste de ejecución de muchos planes de acceso alternativos y eligiendo el que tiene el coste mínimo estimado.

**orden de clasificación.** Secuencia en la que se ordenan los caracteres con el fin de clasificar, fusionar, comparar y procesar datos indexados de forma secuencial.

**orientación de fetch.** Especificación de la colocación deseada del cursor como parte de una sentencia FETCH (por ejemplo BEFORE, AFTER, NEXT, PRIOR, CURRENT, FIRST, LAST, ABSOLUTE y RELATIVE). Véase también “desplazamiento” en la página 1553.

**otorgar.** Dar un privilegio o una autorización a un identificador de autorización.

## P

**página.** (1) Bloque de almacenamiento dentro de una tabla o índice cuyo tamaño es de 4096 bytes (4 KB). (2) Unidad de almacenamiento en un espacio de tabla (4 KB, 8 KB, 16 KB o 32 KB) o un espacio de índice (4 KB). En un espacio de tablas, una página contiene una o más filas de una tabla. En un espacio de tabla LOB, un valor de LOB puede incluir más de una página, pero no se almacena más de un valor de LOB en una página.

**página de códigos.** Conjunto de asignaciones de caracteres para elementos de código.

**página no terminal.** Página que contiene claves y números de página de otras páginas del índice (páginas terminales o no terminales). Las páginas no terminales nunca apuntan a datos reales. Véase también “página terminal” en la página 1588.

**página raíz.** En DB2 Universal Database para OS/390, página de un conjunto de páginas de índice que sigue a la primera página de correlación del espacio de índice. Una página raíz es el nivel más alto (o punto inicial) del índice.

## Glosario

**página terminal.** Página que contiene pares de claves y RID y que apunta a datos reales. Véase también “página no terminal” en la página 1587.

**palabra clave.** (1) Una de las palabras predefinidas de un sistema, lenguaje de mandatos o aplicación. (2) Nombre que identifica una opción utilizada en una sentencia de SQL.

**palabra de clase.** Palabra individual que indica la naturaleza de un atributo de datos.

**palabra reservada.** (1) Palabra que se utiliza en un programa fuente para describir una acción que debe efectuar el programa o compilador. No debe aparecer en el programa como nombre definido por el usuario o nombre del sistema. (2) Palabra que se ha dejado aparte para uso especial en el estándar de SQL.

**panel.** En DB2 Universal Database para OS/390, imagen de pantalla predefinida que define las ubicaciones y características de los campos de pantalla en una superficie de visualización (por ejemplo, un panel de menú).

**paquete.** Estructura de control generada durante la preparación de un programa que se utiliza para ejecutar sentencias de SQL.

**paquete.** En la comunicación de datos, secuencia de dígitos binarios, incluidos los datos y las señales de control, que se transmite y conmuta como un todo compuesto.

**paquete de desencadenante.** Paquete que se crea cuando se ejecuta una sentencia CREATE TRIGGER. El paquete se ejecuta cuando se activa el desencadenante.

**paquete de función.** En DB2 Universal Database para OS/390, paquete que resulta de enlazar el DBRM para un programa de función.

**paquete no operativo.** Paquete que no se puede utilizar porque se han eliminado una función de la que depende. Un paquete de este tipo debe volverse a enlazar lógicamente de forma explícita. Véase también “paquete no válido”.

**paquete no válido.** Paquete que deviene no válido cuando se elimina un objeto del que depende dicho paquete. (El tipo del objeto es distinto de función; por ejemplo, un índice). Un paquete de este tipo se vuelve a enlazar de forma implícita cuando se invoca. Véase también “paquete no operativo”.

**paralelismo.** Posibilidad de realizar múltiples operaciones de base de datos al mismo tiempo en paralelo. Véase también “paralelismo entre particiones” en la página 1589, “paralelismo intrapartición” en la página 1589 y “E/S en paralelo” en la página 1558.

**paralelismo CP de consulta.** En DB2 Universal Database para OS/390, ejecución en paralelo de una consulta individual, que se lleva a cabo utilizando varias tareas. Véase “paralelismo de consulta de Sysplex”.

**paralelismo de consulta de Sysplex.** Ejecución en paralelo de una consulta individual que se realiza utilizando varias tareas en más de un subsistema de DB2 Universal Database para OS/390. Véase también “paralelismo CP de consulta”.

**paralelismo de E/S.** Véase “E/S en paralelo” en la página 1558.

**paralelismo de E/S de consulta.** En DB2 Universal Database para OS/390, acceso en paralelo a datos, que se lleva a cabo activando varias peticiones de E/S dentro de una consulta individual

**paralelismo entre consultas.** Posibilidad de múltiples aplicaciones de consultar una base de datos al mismo tiempo. Cada consulta se ejecuta de forma independiente de las demás, pero DB2 las ejecuta todas al mismo tiempo. Véase también “paralelismo intraconsulta”.

**paralelismo entre particiones.** Operación individual de la base de datos (por ejemplo la creación de índice) que se ejecuta en paralelo entre las particiones de una base de datos particionada. Véase también “paralelismo intrapartición”.

**paralelismo intraconsulta.** Capacidad de procesar al mismo tiempo partes de una consulta individual utilizando el “paralelismo intrapartición”, el “paralelismo entre particiones” o ambos.

**paralelismo intrapartición.** Subdivisión de una sola operación de la base de datos (por ejemplo la creación de índice) en varias partes, que se ejecutan entonces en paralelo en una sola partición de la base de datos. Véase también “paralelismo entre particiones”.

**Parallel Sysplex.** Conjunto de sistemas OS/390 que se comunican entre sí y trabajan conjuntamente mediante determinados componentes de hardware y servicios de software multisistema.

**parámetro de configuración.** Conjunto de limitaciones que se ponen en los valores de los recursos disponibles para una instancia de base de datos o una base de datos.

**parámetro de configuración de base de datos.** (1) Archivo que se crea al crear la instancia del gestor de bases de datos. La mayoría de los parámetros afectan a la cantidad de recursos del sistema que se asignarán a una sola instancia del gestor de bases de datos o configuran la puesta a punto del gestor de bases de datos y los diferentes subsistemas de comunicaciones basándose en consideraciones acerca del entorno. (2) Serie de parámetros que no se pueden cambiar y que son sólo informativos. Todos estos parámetros tienen una aplicabilidad global independiente de cualquier base de datos individual almacenada bajo dicha instancia del gestor de bases de datos.

**parámetros de configuración automáticos.** Conjunto de parámetros de configuración cuyo valor puede cambiar el gestor de bases de datos para reflejar la utilización actual de los recursos.

**parámetros de configuración cambiables.** Conjunto de parámetros de configuración que contienen información que se puede modificar.

**parámetros de configuración en línea cambiables.** Conjunto de parámetros de configuración cuyos valores pueden cambiarse mientras se ejecuta el gestor de bases de datos. Estos cambios entran en vigor inmediatamente.

**par substituto.** En el entorno OS/390, representación codificada de un carácter individual que consta de una secuencia de dos valores Unicode, donde el primer valor del par es un substituto alto en el rango de U+D800 a U+DBFF y el segundo valor es un substituto bajo en el rango de U+DC00 a U+DFFF. Los pares substitutos proporcionan un amplio mecanismo para codificar 917.476 caracteres sin necesidad de utilizar caracteres de 32 bits.

**partición.** En un entorno OS/390, porción de un conjunto de páginas. Cada partición corresponde a un único archivo ampliable independientemente. Las particiones pueden ampliarse a un tamaño máximo de 1, 2 ó 4 gigabytes, en función del número de particiones del conjunto de páginas particionado. Todas las particiones de un conjunto de páginas determinado tienen el mismo tamaño máximo.

## Glosario

**particionamiento hash.** Método de particionamiento en el que se aplica una función hash (de cálculo aleatorio) al valor de clave de particionamiento para determinar la partición de base de datos a la que se asigna la fila.

**partición coordinadora.** Partición a la que se ha conectado originalmente la aplicación y en la que reside el agente de coordinación.

**partición de base de datos.** En un entorno de base de datos particionada, parte de la base de datos que consta de sus propios datos de usuario, índices, archivos de configuración y anotaciones cronológicas de transacciones.

**partición de catálogo.** (1) Partición de base de datos en la que residen las tablas de catálogo. La partición de catálogo puede ser una partición diferente para cada base de datos. (2) En un entorno de base de datos particionada, partición de base de datos donde residen las tablas de catálogo para la base de datos. La partición de catálogo de cada base de datos puede residir en una partición diferente de la instancia.

**partición de datos.** En un entorno OS/390, archivo VSAM que está contenido en un espacio de tablas particionado.

**partición de índice.** Parte de un índice que está asociada con una partición de tabla en una partición de base de datos determinada. Un índice definido en una tabla se implementa mediante múltiples particiones de índice, una por cada partición de tabla.

**partición lógica.** En DB2 Universal Database para OS/390, conjunto de pares clave-RID en un índice sin particionamiento que están asociados a una partición determinada.

**partición lógica de índice.** En DB2 Universal Database para OS/390, conjunto de todas las claves que hacen referencia a la misma partición de datos.

**participante.** En un entorno OS/390, entidad distinta del coordinador de confirmación que interviene en el proceso de confirmación. Sinónimo de *agente* en SNA.

**paso.** En Centro de depósito de datos, operación individual sobre datos perteneciente a un proceso de depósito. En la mayoría de los casos, un paso de proceso comprende una fuente de depósito, una descripción de la transformación o movimiento de datos y un destino. Un paso puede ejecutarse de acuerdo con una planificación o puede realizarse en cascada (“cascada” en la página 1537) desde otro paso.

**paso a través.** En un sistema de bases de datos federado, recurso mediante el cual los usuarios se pueden comunicar con fuentes de datos utilizando el lenguaje SQL de la fuente de datos.

**patrón de búsqueda.** Predicado que puede evaluarse como argumento de búsqueda.

**PCT (program control table).** En CICS, tabla de control de programas.

**PDS.** Véase “conjunto de datos particionado” en la página 1543.

**pendiente.** Estado de una unidad de recuperación que indica que los cambios efectuados por la misma en recursos recuperables de DB2 Universal Database para OS/390 son dudosos y deben aplicarse al soporte de almacenamiento DASD o cancelarse, según determine el coordinador de confirmación.

**pendiente de recuperación de conjunto de páginas (PSRCP).** En DB2 Universal Database para OS/390, estado restrictivo de un espacio de índice en el cual se debe recuperar el conjunto de páginas completo. La recuperación de una parte lógica está prohibida.

**pendiente de recuperación lógica (LRECP).** En DB2 Universal Database para OS/390, estado en el cual los datos y las claves de índice que hacen referencia a los datos son incoherentes.

**pendiente de reinicio (RESTP).** En DB2 Universal Database para OS/390, estado restrictivo de un conjunto de páginas o partición que indica que debe efectuarse un trabajo de reinicio (retrotracción) en el objeto. Se rechaza todo el acceso al conjunto de páginas o partición, salvo el acceso mediante el mandato RECOVER POSTPONED o mediante la retrotracción en línea automática, que DB2 Universal Database para OS/390 invoca después del reinicio si el parámetro del sistema LBACKOUT es igual a AUTO.

**pendiente de REORG (REORP).** En DB2 Universal Database para OS/390, condición que limita el acceso de SQL y la mayor parte del acceso de los programas de utilidad a un objeto que debe reorganizarse.

**perfil de cliente.** Perfil usado para configurar clientes utilizando la función de importación (Import) en el Client Configuration Assistant. Puede contener información de conexión de base de datos, valores de cliente, parámetros comunes de CLI u ODBC y datos de configuración para subsistemas de comunicaciones APPC o NetBIOS locales. Véase también “perfil de servidor”.

**perfil de entorno.** Script que se proporciona con Text Extender que contiene valores para variables de entorno.

**perfil de información complementaria de CPI-C.** En SNA, perfil que especifica las características de conversación que se deben utilizar al asignar una conversación con un programa de transacciones remoto. Este perfil es utilizado por los programas de transacciones locales que se comunican mediante CPI-C (interfaz común de programación para comunicaciones). El perfil especifica el nombre de LU remota (el nombre del perfil de conexión donde está el nombre de LU remota), el nombre de modalidad y el nombre del programa de transacciones remoto.

**perfil de seguridad de conversación.** Conjunto de identificadores de usuario o de identificadores de grupo y contraseñas que utiliza APPC para la seguridad de la conversación.

**perfil de servidor.** Perfil que contiene información acerca de las instancias de servidor de un sistema y de las bases de datos de cada instancia de servidor. Véase también “perfil de cliente”.

**perfil de variable de rendimiento.** Archivo plano que contiene definiciones de variables de rendimiento. Este archivo se puede editar, copiar y compartir. Los diferentes perfiles pueden ser utilizados por el mismo Supervisor de rendimiento para poder efectuar cálculos diferentes.

**persistencia.** En Net.Data, estado de mantener un valor asignado para una transacción completa, cuando una transacción abarca varias invocaciones de Net.Data. Sólo pueden ser persistentes las variables. Además, las operaciones sobre recursos afectados por el control de confirmación se mantienen activas hasta que se realiza una confirmación o retrotracción, o hasta que finaliza la transacción.

**petionario.** (1) Origen de una petición para acceder a datos en un servidor remoto. Asimismo, el sistema que solicita los datos. Para DB2 Universal Database para OS/390, la función de petionario la proporciona el recurso de datos distribuidos para acceder a un RDBMS remoto. En función del nivel de protocolo DRDA utilizado, un petionario puede describirse como un petionario de aplicaciones

## Glosario

(“petionario de aplicaciones”) o un servidor de bases de datos (“servidor de bases de datos” en la página 1608 ). (2) Destino de una petición de un petionario remoto.

**petionario de aplicaciones.** Componente de un sistema remoto que genera peticiones DRDA de datos en nombre de la aplicación. Un petionario de aplicaciones accede a un servidor de bases de datos DB2 utilizando el protocolo dirigido por aplicación DRDA. Véase también “servidor de aplicaciones” en la página 1607.

**petición de conexión remota.** En DB2 Universal Database para OS/390, petición efectuada por una ubicación remota para conectarse al subsistema DB2 local. Específicamente, la petición enviada es una cabecera 5 de gestión de función de SNA.

**petición distribuida.** En un sistema de bases de datos federado, consulta de SQL dirigida hacia dos o más fuentes de datos.

**pieza.** En un entorno OS/390, conjunto de datos de un conjunto de páginas particionado.

**pila.** Área de la memoria que almacena información temporal de registros, parámetros y direcciones de retorno de subrutinas.

**plan.** Véase “plan de aplicación”.

**plan de acceso.** Conjunto de vías de acceso que el optimizador selecciona para evaluar una sentencia de SQL determinada. El plan de acceso especifica el orden de las operaciones para determinar el plan de ejecución, los métodos de implementación (por ejemplo, JOIN) y la vía de acceso de cada tabla mencionada en la sentencia.

**plan de aplicación.** Estructura de control que se crea durante el proceso de enlace lógico. DB2 Universal Database para OS/390 utiliza el plan de aplicación para procesar las sentencias de SQL que encuentra durante la ejecución de sentencias.

**planificador de trabajos.** Programa utilizado para automatizar determinadas tareas con el fin de ejecutar y gestionar trabajos de base de datos.

**plantilla.** En el entorno OS/390, descriptor de conjunto de datos de salida de programas de utilidad de DB2 que se utiliza para la asignación dinámica. Una plantilla se define mediante la sentencia de control de programa de utilidad TEMPLATE.

**plantilla de función.** En una base de datos federada, función parcial que no tiene código ejecutable. El usuario correlaciona la plantilla de función con una función de fuente de datos, de forma que ésta se pueda invocar desde el servidor federado.

**PLT (program list table).** En CICS, tabla de lista de programas.

**política.** Véase “política CFRM”.

**política CFRM.** En DB2 Universal Database para OS/390, declaración efectuada por un administrador de MVS con respecto a las normas de asignación de una estructura de recurso de acoplamiento.

**posibilidad de actualización.** Posibilidad de un cursor de realizar actualizaciones y supresiones colocadas. La posibilidad de actualización de un cursor puede estar influida por la sentencia SELECT y la opción de sensibilidad del cursor que se especifique en la sentencia DECLARE CURSOR.

**PPT.** (1) En CICS, programa de proceso. (2) En OS/390, tabla de propiedades de programa.

**precaptación secuencial.** Mecanismo que desencadena operaciones de E/S asíncronas consecutivas. Las páginas se cargan antes de que sean necesarias y se leen varias páginas en una sola operación de E/S.

**precisión.** En los tipos de datos numéricos, número total de dígitos binarios o decimales, excluido el signo. El signo se considera positivo si el valor de un número es cero.

**precompilar.** Procesar programas que contienen sentencias de SQL antes de compilarlos. Las sentencias de SQL se sustituyen por sentencias que serán reconocidas por el compilador del lenguaje principal. La salida de un proceso de precompilación incluye código fuente que se puede someter al compilador y utilizar en el proceso de enlace de programas.

**predicado.** Elemento de una condición de búsqueda que expresa o implica una operación de comparación.

**predicado básico.** Predicado que compara dos valores.

**predicado cuantificado.** Predicado que compara un valor con un conjunto de valores.

**predicados de búsqueda por índice.** Predicados de SQL que se aplican a entradas de índice, en páginas terminales de índice, para reducir el número de entradas de índice que califican la petición de SQL. Estos predicados ayudan a disminuir el número de filas de datos accedidas.

**prefijo de mandato.** En DB2 Universal Database para OS/390, identificador de mandato que tiene de 1 a 8 caracteres. El prefijo de mandato permite diferenciar si el mandato pertenece a una aplicación o subsistema, en lugar de a OS/390.

**preparar.** (1) Convertir una sentencia de SQL del formato de texto a un formato ejecutable, sometiéndola al compilador de SQL. (2) Primera fase de un proceso de confirmación de dos fases en la cual se solicita a todos los participantes que se preparen para la confirmación.

**principal.** Entidad que puede comunicarse de forma segura con otra entidad. En Kerberos, los principales se representan como entradas de la base de datos de registro de Kerberos e incluyen usuarios, servidores, sistemas y otros elementos.

**privilegio.** (1) Derecho a tener acceso a un objeto específico de la base de datos de un modo específico. Estos derechos los controlan los usuarios con autorización SYSADM (administrador del sistema), autorización DBADM (administrador de bases de datos) o los creadores de los objetos. Por ejemplo, derechos tales como crear, suprimir y seleccionar datos de las tablas. (2) En DB2 Universal Database para OS/390, posibilidad de llevar a cabo una función específica, a veces en un objeto específico. Véase también "privilegio explícito" y "privilegio implícito" en la página 1594.

**privilegio de control.** Autorización para controlar completamente un objeto. Esto incluye la autorización para acceder, eliminar o modificar un objeto y la autorización para ampliar o revocar privilegios sobre el objeto a otros usuarios.

**privilegio de propiedad.** Privilegio de CONTROL que permite todos los privilegios para el objeto de datos de propiedad. Véase "privilegio".

**privilegio explícito.** Privilegio que tiene un nombre y se ostenta como resultado de sentencias GRANT y REVOKE de SQL, por ejemplo, el privilegio SELECT. Véase "privilegio". Véase también "privilegio implícito" en la página 1594.



## Glosario

**privilegio implícito.** (1) Privilegio asociado a la posesión de un objeto, tal como el privilegio para eliminar un sinónimo que es propiedad del usuario o la posesión de una autorización, tal como el privilegio de la autorización SYSADM para interrumpir cualquier trabajo de programa de utilidad. (2) Se otorga a un usuario que tiene el privilegio de ejecutar un paquete en objetos de datos utilizados en el paquete que no necesitan que se les otorgue privilegios explícitos. Véase “privilegio” en la página 1593. Véase también “privilegio explícito” en la página 1593.

**privilegio individual.** Privilegio otorgado en un objeto de datos individual. Véase “privilegio” en la página 1593.

**procedimiento.** Véase “procedimiento almacenado”.

**procedimiento almacenado.** (1) Programa de aplicación, que posiblemente contiene sentencias de SQL, que se puede invocar con la sentencia CALL de SQL. (2) Programa de aplicación escrito por el usuario que se puede iniciar mediante el uso de la sentencia CALL de SQL.

**procedimiento de campo.** En DB2 Universal Database para OS/390, rutina de salida escrita por el usuario que está diseñada para aceptar un valor individual y transformarlo (codificarlo o decodificarlo) del modo que especifique el usuario.

**procedimiento externo.** Programa de aplicación escrito en un lenguaje principal, que posiblemente contiene sentencias de SQL, que se puede invocar con la sentencia de SQL CALL. Véase también “procedimiento SQL”.

**procedimiento SQL.** Programa de aplicación escrito en SQL que se puede invocar con la sentencia CALL de SQL. Véase también “procedimiento externo”.

**Procesador de línea de mandatos (CLP).** Interfaz basada en caracteres que sirve para entrar sentencias de SQL y mandatos del gestor de bases de datos.

**proceso.** (1) En Centro de depósito de datos, serie de pasos, que normalmente actúa sobre datos fuente, y que convierte datos de su forma original a una forma propicia para el soporte de decisiones. Un proceso del Centro de depósito de datos consta habitualmente de una o más fuentes, uno o más pasos y uno más destinos. (2) Unidad a la que el gestor de bases de datos asigna recursos y bloqueos. Un proceso que incluye la ejecución de uno o más programas. La ejecución de una sentencia de SQL está siempre asociada con un proceso. La forma de iniciar y la forma de finalizar un proceso dependen del entorno. Sinónimo de “proceso de aplicación”.

**proceso analítico en línea (online analytical processing - OLAP).** En el Kit de iniciación de OLAP, entorno de proceso cliente/servidor, multidimensional y multiusuario, diseñado para los usuarios que necesitan analizar datos comerciales globales en tiempo real.

**proceso de aplicación.** Unidad a la que se asignan recursos y bloqueos. Un proceso de aplicación incluye la ejecución de uno o más programas.

**proceso de notificación.** Proceso creado por el Centro de depósito de datos que contiene todos los pasos de proceso creados para notificación cuando finaliza un paso de proceso.

**proceso de publicación de metadatos.** Proceso creado por el Centro de depósito de datos que contiene todos los pasos para mantener los metadatos publicados sincronizados con los metadatos originales.

**proceso de registro.** En el proceso de duplicación de datos de DB2, proceso de definir una fuente de duplicación. Véase también “proceso de suscripción” en la página 1595.



**proceso de suscripción.** En el proceso de duplicación de datos de DB2, proceso en el que el usuario define conjuntos de suscripción y miembros de conjunto de suscripción. Véase también “proceso de registro” en la página 1594.

**proceso masivamente paralelo (MPP).** Más de un sistema de uniprosesor o multiprosesor simétrico (SMP) enlazados mediante una red de alta velocidad.

**proceso paralelo de E/S.** Modalidad de proceso de E/S en la que DB2 Universal Database para OS/390 inicia varias peticiones simultáneas para una consulta individual del usuario y ejecuta el proceso de E/S simultáneamente (en paralelo) en varias particiones de datos.

**programa Apply.** En el proceso de duplicación de datos de DB2, programa que se utiliza para renovar o actualizar una tabla destino, de acuerdo con las reglas que rigen la relación fuente-destino. Véase también “programa Capture”, “desencadenante de Capture” en la página 1552, “tabla de destino” en la página 1615 y “tabla fuente” en la página 1616.

**programa Capture.** En el proceso de duplicación de datos de DB2, programa que lee registros de anotaciones de base de datos o de diario para captar datos acerca de cambios efectuados en tablas fuente de DB2. Véase también “programa Apply” y “desencadenante de Capture” en la página 1552.

**programación del extremo del servidor.** Método para añadir datos de DB2 en páginas Web dinámicas. Common Gateway Interface (CGI), los programas de API de servidor Web y las servlets Java son tres tipos comunes de programas del extremo del servidor.

**programa definido por el usuario.** Programa que un usuario proporciona y define para el Centro de depósito de datos, a diferencia de los programas proporcionados con el Centro de depósito de datos, que se definen automáticamente.

**programa de instalación.** Programa que prepara un paquete de software para ejecutarlo en el sistema. Generalmente, durante la instalación, se copia y se deja un componente del programa de configuración en la unidad de disco duro para permitir al usuario personalizar los valores por omisión del programa.

**programa del Centro de depósito de datos.** Programa, proporcionado con el Centro de depósito de datos, que se puede iniciar desde el Centro de depósito de datos y que se define automáticamente, por ejemplo, los programas de carga y transformadores de DB2.

**programa de lenguaje principal.** Programa escrito en un lenguaje principal que contiene sentencias de SQL intercaladas.

**programa de proceso de mensajes (MPP).** En un entorno OS/390 con IMS, programa de proceso de mensajes. Por ejemplo, los programas de aplicación que contienen sentencias de SQL se ejecutan en el programa de proceso de mensajes (MPP), el programa de mensajes de proceso por lotes (BMP), las regiones de Vía de acceso rápida o las regiones de proceso por lotes IMS.

**programa de transacciones (TP).** Programa de aplicación que utiliza APPC para comunicarse con un programa de aplicación asociado.

**Programa de utilidad DBA.** Herramienta que permite a los usuarios de DB2 configurar bases de datos e instancias del gestor de bases de datos, gestionar los directorios necesarios para acceder a bases de datos locales y remotas, copiar y recuperar bases de datos o espacios de tablas y gestionar soportes de almacenamiento en un sistema utilizando una interfaz gráfica. Se puede acceder a las tareas proporcionadas por esta herramienta desde el Centro de control.

## Glosario

**programa de utilidad db2setup.** Programa de utilidad utilizado para guiar a los usuarios durante el proceso de instalación utilizando una interfaz y ayuda en línea. Este programa de utilidad de configuración puede crear o asignar grupos e ID de usuarios, crear una instancia de DB2 e instalar mensajes del producto. Se proporcionan valores por omisión para todos los parámetros de instalación necesarios.

**programa de utilidad de carga.** Programa de utilidad no transaccional que efectúa actualizaciones por bloques de datos de tabla. Véase también “programa de utilidad de importación” y “programa de utilidad de exportación”.

**programa de utilidad de exportación.** Programa de utilidad con transacciones que extrae datos de una tabla. Véase también “programa de utilidad de importación” y “programa de utilidad de carga”.

**programa de utilidad de importación.** Programa de utilidad de transacciones que inserta en una tabla datos de registro proporcionados por el usuario. Véase también “programa de utilidad de carga” y “programa de utilidad de exportación”.

**programa fuente.** Conjunto de sentencias de lenguaje principal y de sentencias de SQL que son procesadas por un precompilador de SQL.

**programas de utilidad regulados.** Programas de utilidad que tienen puesto un límite en los recursos que, de lo contrario, consumirían. El grado al que están limitados los recursos se basa en la carga de trabajo actual del sistema. Los programas de utilidad soportados incluyen la copia de seguridad, la restauración y la reorganización de espacios de tablas.

**promoción de bloqueos.** Proceso de cambiar el tamaño o la modalidad de un bloqueo de DB2 Universal Database para OS/390 a un nivel superior.

**propagación.** Grupos de parámetros de configuración que se actualizan y entran en vigor a velocidades diferentes.

**propiedad.** En el Centro de depósito de datos, característica o atributo que describe una unidad de información. Cada tipo de objeto tiene un conjunto de propiedades asociadas. Para cada objeto, se asigna un conjunto de valores a las propiedades.

**propiedad del Centro de depósito de datos.** Atributo que se aplica entre sesiones del Centro de depósito de datos, por ejemplo la base de datos de control de depósito que contiene los metadatos técnicos. Véase también “propiedad”.

**propiedad de objeto.** Propiedad que identifica una categoría de información que está asociada a un objeto. Se pueden asignar una o más propiedades a un objeto de base de seguridad de NetWare. El objeto de instancia del servidor de DB2 tiene una propiedad de objeto, NET\_ADDR, que indica la ubicación del registro dentro del objeto.

**propietario del paquete de función.** En DB2 Universal Database para OS/390, identificador de autorización del usuario que enlaza el DBRM del programa de función en un paquete de función.

**protocol.ini.** Archivo que contiene información de enlace lógico y configuración de la LAN para todos los módulos de protocolo y del sistema de control de acceso al medio (MAC).

**protocolos de sesión.** En DB2 Universal Database para OS/390, conjunto disponible de peticiones y respuestas de comunicaciones SNA.

**proyecto del Generador de Procedimientos Almacenados.** Archivo, creado por el Generador de Procedimientos Almacenados, que contiene información sobre conexiones y objetos de procedimientos almacenados que no se han creado satisfactoriamente en la base de datos.

**PSRCP (page set recovery pending).** Véase “pendiente de recuperación de conjunto de páginas” en la página 1591.

**PU.** Véase “unidad física” en la página 1621.

**puerto TCP/IP.** Valor de 2 bytes que identifica a un usuario final o a una aplicación de red TCP/IP dentro de un sistema principal TCP/IP.

**punto de coherencia.** Momento determinado en el que todos los datos recuperables a los que tiene acceso un programa son coherentes. El punto de coherencia se produce cuando las actualizaciones, inserciones y supresiones se confirman en la base de datos física o se retrotraen. Sinónimo de “punto de confirmación”. Véase también “retrotracción” en la página 1604.

**punto de confirmación.** Momento en el que se considera que los datos son coherentes.

**punto de control.** Punto en el que el gestor de bases de datos registra información de estado interna en la anotación cronológica; el proceso de recuperación utiliza esta información si el subsistema termina anormalmente.

**punto de control.** (1) En APPN, componente de un nodo que gestiona los recursos de dicho nodo y proporciona opcionalmente servicios a otros nodos de la red. Por ejemplo, un punto de control de servicios del sistema (SSCP) en un nodo de tipo 5, un punto de control de unidad física (PUCP) en un nodo de tipo 4, un punto de control de nodo de red (NNCP) en un nodo de red de tipo 2.1 (T2.1) y un punto de control de nodo final (ENCP) en un nodo final T2.1. Un SSCP y un NNCP pueden proporcionar servicios a otros nodos. (2) Componente de un nodo T2.1 que gestiona los recursos de dicho nodo. Si el nodo T2.1 es un nodo APPN, el punto de control puede participar en sesiones de punto de control a punto de control con otros nodos APPN. Si el nodo T2.1 es un nodo de red, el punto de control también proporciona servicios a nodos finales adyacentes de la red T2.1. Véase también “unidad física” en la página 1621 y “nombre de punto de control” en la página 1584.

**punto de control de servicios del sistema (SSCP).** Punto de control en una red SNA que proporciona servicios de red para nodos dependientes.

**punto de control por software.** Proceso de escribir información en la cabecera del archivo de anotaciones cronológicas; esta información se utiliza para determinar el punto inicial del archivo de anotaciones por si fuera necesario efectuar un reinicio de la base de datos.

**punto de inmovilización.** Punto en el que los datos son coherentes como resultado de la ejecución del programa de utilidad QUIESCE de DB2.

**punto de sincronismo.** Véase “punto de coherencia”.

**punto muerto.** Condición bajo la cual una transacción no puede continuar porque es dependiente de recursos exclusivos que están bloqueados por otra transacción, que a su vez es dependiente de recursos exclusivos que están siendo utilizados por la transacción original.

## Glosario

### Q

**QBIC.** Véase “Query by Image Content”.

**QSAM (queued sequential access method).** Véase “método de acceso secuencial en cola” en la página 1580.

**Query by Image Content (Consulta por contenido de imagen - QBIC).** Capacidad del Image Extender que permite al usuario buscar imágenes de acuerdo con sus características visuales, tales como el promedio de color y textura.

### R

**RACF.** Véase “Resource Access Control Facility” en la página 1603.

**RAMAC.** En un entorno OS/390, familia de productos de IBM de almacenamiento de disco para la empresa.

**rango de páginas erróneas.** Rango de páginas que se consideran físicamente dañadas. DB2 Universal Database para OS/390 no permite que los usuarios accedan a ninguna página que esté dentro de este rango.

**rastreo.** Recurso de DB2 Universal Database para OS/390 que permite supervisar y reunir datos (globales) de supervisión, auditoría, rendimiento, contabilidad, estadísticas y disponibilidad referentes a DB2 Universal Database para OS/390.

**RBA.** Véase “relative byte address” en la página 1554.

**RCT (resource control table).** En DB2 Universal Database para OS/390 junto con el recurso de conexión CICS, es la tabla de control de recursos.

**RDB.** Véase “base de datos relacional” en la página 1533.

**RDBMS.** Véase “sistema de gestión de bases de datos relacionales” en la página 1610.

**RDBNAM.** Véase “nombre de base de datos relacional” en la página 1583.

**RDF (record definition field).** En DB2 Universal Database para OS/390, campo de definición de registro.

**reajuste de bloqueos.** En el gestor de bases de datos, respuesta que se produce cuando el número de bloqueos emitidos para un agente excede el límite especificado en la configuración de la base de datos; el límite se define mediante el parámetro de configuración *maxlocks*. Durante un reajuste de bloqueos, se liberan bloqueos convirtiendo los bloqueos sobre las filas de una tabla en un bloqueo individual sobre la tabla. Esto se repite hasta que ya no se exceda el límite.

**rearranque condicional.** Rearranque de DB2 Universal Database para OS/390 que está dirigido por un registro de control de rearranque condicional (CRCR) definido por el usuario.

**rechazo en cascada.** En el proceso de duplicación de datos de DB2, proceso de rechazar una transacción de duplicación porque está asociada a una transacción en la que se detectó un conflicto y fue ella misma rechazada.

**reclamación.** En DB2 Universal Database para OS/390, notificación al gestor de base de datos para indicar que se está accediendo a un objeto. La reclamación evita que se produzcan drenajes hasta que se libere la reclamación, lo cual ocurre habitualmente en un punto de confirmación. Véase también “drenar” en la página 1555.

**reclamación física.** En DB2 Universal Database para OS/390, reclamación sobre un índice completo sin particionamiento.

**reclamación lógica.** En DB2 Universal Database para OS/390, reclamación sobre una partición lógica de un índice sin particionamiento.

**reconstrucción del estado actual.** En DB2 Universal Database para OS/390, segunda fase del proceso de rearranque durante la cual se reconstruye el estado del subsistema a partir de información del archivo de anotaciones.

**Recoverable Resource Manager Services (Servicios del gestor de recursos recuperables - RRSF).** Recurso de conexión de Recoverable Resource Manager Services, que es un subcomponente de DB2 Universal Database para OS/390 que utiliza los OS/390 Transaction Management and Recoverable Resource Manager Services para coordinar la confirmación de recursos entre DB2 Universal Database para OS/390 y todos los demás gestores de recursos que también utilizan OS/390 RRS en un sistema OS/390.

**RECP (recovery pending).** En DB2 Universal Database para OS/390, recuperación pendiente.

**recuperación.** Proceso de reconstrucción de una base de datos que ha quedado inutilizable debido a una anomalía de hardware y/o software. El proceso incluye la restauración de una imagen de copia de seguridad y puede incluir la recuperación en avance de tiempo de las anotaciones cronológicas de base de datos.

**recuperación ascendente mediante archivo de anotaciones.** Cuarta y última fase del proceso de reinicio durante la cual DB2 Universal Database para OS/390 explora el archivo de anotaciones en sentido regresivo para aplicar registros UNDO para todos los cambios cancelados.

**recuperación ascendente mediante archivo de anotaciones.** Tercera fase del proceso de reinicio durante la cual DB2 Universal Database para OS/390 procesa el archivo de anotaciones en sentido ascendente para aplicar todos los registros REDO del archivo.

**recuperación de anomalía.** Proceso de recuperación de una anomalía inmediata. Véase también “recuperación de versión” y “recuperación en avance” en la página 1600.

**recuperación de desastre.** Actividades que es necesario realizar para restaurar la base de datos en el caso de incendio, terremoto, vandalismo u otros sucesos catastróficos. Normalmente, para realizar la recuperación de desastre es necesario restaurar la base de datos entera, de forma que cuando se produce un desastre importante, se necesita una copia de seguridad de la base de datos entera en un sitio de espera.

**recuperación de versión.** Restauración de una versión anterior de la base de datos, utilizando una imagen que se ha creado durante una operación de copia de seguridad. Véase “recuperación de anomalía”. Véase también “recuperación en avance” en la página 1600.

## Glosario

**recuperación en avance.** Proceso utilizado para volver a crear una base de datos o un espacio de tablas restaurado con el estado que tenían en un momento determinado aplicando los cambios registrados en la “anotación cronológica de base de datos” en la página 1528.

**recuperación pendiente.** Estado de la base de datos o del espacio de tablas. Una base de datos o un espacio de tablas queda en estado de pendiente de recuperación cuando se restaura desde una copia de seguridad. Mientras la base de datos o el espacio de tablas está en este estado, no se puede tener acceso a sus datos.

**recurso.** En DB2 Universal Database para OS/390, el objeto de un bloqueo o reclamación, que puede ser un espacio de tablas, un espacio de índice, una partición de datos, una partición de índice o una partición lógica.

**recurso de acoplamiento.** En un entorno OS/390, partición lógica LPAR de PR/SM™ designada que ejecuta el programa de control del recurso de acoplamiento y proporciona almacenamiento en antememoria de alta velocidad, proceso de listas y funciones de bloqueo en un Parallel Sysplex.

**recurso de acoplamiento entre sistemas (cross-system coupling facility - XCF).** Componente de OS/390 que proporciona funciones para dar soporte a la cooperación entre programas autorizados que se ejecutan dentro de un Parallel Sysplex.

**recurso de auditoría.** Programa de utilidad que genera un seguimiento de registros de auditoría para una serie de sucesos predefinidos y supervisados de la base de datos.

**recurso de conexión.** Interfaz entre DB2 Universal Database para OS/390 y TSO, IMS, CICS o espacios de direcciones de proceso por lotes. El recurso de conexión permite que los programas de aplicación accedan a DB2 Universal Database para OS/390.

**recurso de conexión de CICS.** Subcomponente de DB2 Universal Database para OS/390 que utiliza la interfaz de subsistema (SSI) de MVS y el enlace entre almacenamientos para procesar peticiones de CICS a DB2 Universal Database para OS/390 y coordinar la confirmación de recursos.

**recurso de conexión de IMS.** Subcomponente de DB2 Universal Database para OS/390 que utiliza los protocolos de interfaz de subsistema (SSI) OS/390 y el enlace entre memorias para procesar peticiones de IMS a DB2 Universal Database para OS/390 y para coordinar la confirmación de recursos.

**recurso de conexión de llamada (call attachment facility - CAF).** Recurso de conexión de DB2 Universal Database para OS/390 para programas de aplicación que se ejecutan en proceso por lotes MVS o TSO. CAF es una alternativa al procesador de mandatos DSN y proporciona un mayor control sobre el entorno de ejecución.

**recurso de conexión de TSO.** Recurso de DB2 Universal Database para OS/390 que consta del procesador de mandatos DSN y DB2I. Las aplicaciones que no están escritas para los entornos CICS ni IMS se pueden ejecutar bajo el recurso de conexión de TSO.

**recurso de datos distribuidos (distributed data facility - DDF).** Conjunto de componentes de DB2 Universal Database para OS/390 mediante los cuales DB2 Universal Database para OS/390 se comunica con otro RDBMS.

**recurso del gestión del sistema (system management facility - SMF).** DB2 Universal Database para OS/390 utiliza SMF para las estadísticas, la información de contabilidad y los datos de rendimiento.

**recurso de límite de recursos (resource limit facility - RLF).** Porción del código de DB2 Universal Database para OS/390 que impide que las sentencias de manipulación de SQL dinámico excedan los límites de tiempo especificados. También se conoce como *rutina de gobierno*.

**recurso de programa autorizado (authorized program facility - APF).** En DB2 Universal Database para OS/390, recurso que permite la identificación de los programas que tienen autorización para utilizar funciones restringidas.

**recurso de rastreo generalizado (generalized trace facility - GTF).** En un entorno OS/390, programa de servicio que registra sucesos importantes del sistema, tales como interrupciones de E/S, interrupciones de SVC, interrupciones de programa o interrupciones externas.

**recurso de recuperación ampliado (extended recovery facility - XRF).** En un entorno OS/390, recurso que minimiza el efecto de errores en MVS, VTAM, el procesador principal o en aplicaciones de alta disponibilidad durante sesiones entre aplicaciones de alta disponibilidad y terminales seleccionados. Este recurso proporciona un subsistema alternativo para hacerse cargo de las sesiones del subsistema anómalo.

**Red APPN.** Conjunto de nodos de red interconectados y sus nodos clientes finales.

**Red avanzada de igual a igual (APPN).** Extensión de SNA que incluye control distribuido de la red, definición dinámica de recursos de red, y registro de recursos y búsqueda de directorios automáticos. Véase también “Arquitectura de red de sistemas” en la página 1531.

**red SNA.** Parte de la red de aplicaciones de usuario que se ajusta a los formatos y protocolos de la Arquitectura de red de sistemas (SNA). Permite transferir datos entre usuarios de forma fiable y proporciona protocolos para controlar los recursos de diversas configuraciones de red. La red SNA consta de unidades direccionables de red (NAU), la función de pasarela, componentes de función de direccionamiento de sesión intermedios y la red de transporte.

**reenlace automático.** Proceso mediante el cual las sentencias de SQL se enlazan automáticamente (sin que un usuario emita un mandato BIND) cuando un proceso de aplicación empieza la ejecución y el plan o paquete de aplicación enlazado que es necesario no es válido. Véase también “enlace” en la página 1557 y “reenlazar”.

**reenlazar.** Creación de un paquete para un programa de aplicación que se enlazó previamente. Por ejemplo, si se añade un índice para una tabla a la que accede un programa, se debe volver a enlazar el paquete para que pueda beneficiarse del nuevo índice. Véase también “reenlace automático” y “enlace” en la página 1557.

**referencia correlacionada.** Referencia a una columna de una tabla que está fuera de una subconsulta.

**registro.** Representación en el almacenamiento de una fila individual de una tabla u otros datos.

**registro de anotaciones.** Registro de una actualización efectuada en una base de datos durante una unidad de trabajo. Este registro se escribe continuación de la cola del archivo de anotaciones activo.

**registro de desbordamiento.** (1) En un archivo direccionado indirectamente, registro cuya clave se genera aleatoriamente en la dirección de una pista completa o en la dirección de un registro inicial. (2) En DB2, registro actualizado que es demasiado grande para incluirse en la página en la que está almacenado actualmente. El registro se copia en una página diferente y su ubicación original se sustituye por un puntero que apunta a la nueva ubicación. (3) En el supervisor de sucesos, registro insertado en la



## Glosario

corriente de datos del supervisor de sucesos para indicar que se han eliminado registros porque el área de interconexión con nombre estaba llena y los registros no se han procesado a tiempo. Un registro de desbordamiento indica cuántos registros se han eliminado.

**registro especial.** Área de almacenamiento que el gestor de bases de datos define para un proceso de aplicación y que se utiliza para almacenar información a la que se puede hacer referencia en sentencias de SQL. USER y CURRENT DATE son ejemplos de registros especiales.

**regla de actualización.** Condición impuesta por el gestor de bases de datos que debe cumplirse para poder actualizar una columna.

**regla de inserción.** Condición aplicada por el gestor de bases de datos que debe cumplirse para poder insertar una fila en una tabla.

**regla de supresión.** Regla asociada a una restricción de referencia que restringe la supresión de una fila padre o especifica el efecto de dicha supresión en las filas dependientes.

**regresión.** (1) Proceso mediante el cual un servidor de bases de datos vuelve automáticamente a ejecutarse en la máquina original cuando ésta queda disponible, después de que una anomalía le haya hecho ejecutarse en otra máquina. (2) Proceso de volver a un release anterior de DB2 Universal Database para OS/390 después de intentar o finalizar la migración a un release actual.

**rehacer.** En DB2 Universal Database para OS/390, estado de una unidad de recuperación que indica que los cambios deben aplicarse de nuevo al soporte de almacenamiento DASD para asegurar la integridad de los datos.

**reiniciador.** En un sistema de bases de datos federado, mecanismo mediante el cual el servidor federado invoca rutinas para comunicarse con una fuente de datos y recuperar datos de ella. Las rutinas están contenidas en una biblioteca denominada *módulo del reiniciador*.

**reinicio de grupo.** En un entorno OS/390, reinicio de como mínimo un miembro de un grupo de compartimiento de datos después de perder los bloqueos o el área de comunicaciones compartidas.

**relación.** En DB2 Universal Database para OS/390, conexión definida existente entre las filas de una tabla o las filas de dos tablas. Una relación es la representación interna de una restricción referencial.

**Relational Connect.** Relational Connect es un producto necesario para los usuarios que desean acceder a otros DBMS, por ejemplo Oracle, Sybase y Microsoft SQL Server.

**remigración.** Proceso de volver a un release actual de DB2 Universal Database después de revertir a un release anterior. Este procedimiento constituye otro proceso de migración.

**remoto.** En DB2 Universal Database para OS/390, cualquier objeto que es mantenido por un subsistema DB2 remoto. Una vista remota, por ejemplo, es una vista mantenida por un subsistema DB2 remoto. Véase también "local" en la página 1577.

**renovación.** Proceso en el que todos los datos de interés de una tabla de usuario se copian en la tabla destino, sustituyendo los datos existentes. Véase también "renovación completa" y "renovación diferencial" en la página 1603.

**renovación completa.** En el proceso de duplicación de datos de DB2, proceso en el que se copian todos los datos de interés de una tabla de usuario en la tabla destino, sustituyendo a los datos existentes. Véase "renovación". Véase también "renovación diferencial" en la página 1603.



**renovación diferencial.** En el proceso de duplicación de datos de DB2, proceso en el que sólo se copian datos modificados en la tabla destino, sustituyendo a los datos existentes. Véase “renovación” en la página 1602. Véase también “renovación completa” en la página 1602.

**reoptimización.** Proceso de DB2 Universal Database para OS/390 de reconsiderar la vía de acceso de una sentencia de SQL en el momento de la ejecución; durante la reoptimización, DB2 Universal Database para OS/390 utiliza los valores de variables de lenguaje principal, marcadores de parámetro o registros especiales.

**REORP.** Véase “pendiente de REORG” en la página 1591.

**resolución de función.** Proceso interno del gestor de bases de datos por el que se selecciona una instancia de función determinada para su invocación. El nombre de función, los tipos de datos de los argumentos y la vía de acceso de la función se utilizan para efectuar la selección. Sinónimo de “selección de función” en la página 1605.

**resolución del estado dudoso.** Proceso de resolución del estado de una unidad de trabajo lógica dudosa al estado de confirmada o retrotraída.

**Resource Access Control Facility (RACF).** Resource Access Control Facility protege el sistema proporcionando acceso a aquellas personas que tienen autorización para utilizar el recurso. RACF es un componente de SecureWay Security Server para OS/390.

**restauración en línea.** Restauración de una copia de una base de datos o un espacio de tablas que se efectúa mientras otras aplicaciones están accediendo a la base de datos o al espacio de tablas. Véase también “restauración fuera de línea”.

**restauración fuera de línea.** Restauración de una copia de una base de datos o un espacio de tablas desde una copia de seguridad. El programa de utilidad de restauración de base de datos tiene el uso exclusivo de la base de datos hasta que se ha completado la restauración. Véase también “restauración en línea”.

**restaurar.** Volver a crear una base de datos dañada o corrupta a partir de una imagen de copia de seguridad.

**RESTP.** Véase “pendiente de reinicio” en la página 1591.

**restricción.** Regla que limita los valores que se pueden insertar, suprimir o actualizar en una tabla. Véase también “restricción de comprobación”, “restricciones de referencia” en la página 1604, “restricción de comprobación de tabla” y “restricción de unicidad” en la página 1604.

**restricción de autorreferencia.** Restricción de referencia que define una relación en la que una tabla es dependiente de sí misma.

**restricción de comprobación.** Restricción que especifica una condición de control que no es falsa para cada fila de la tabla donde se ha definido la restricción. Véase “restricción de comprobación de tabla”.

**restricción de comprobación de tabla.** Restricción definida por el usuario que especifica los valores que pueden contener las columnas específicas de una tabla base.

**restricción de referencia cíclica.** Tabla que es dependiente o descendiente de otra tabla.

## Glosario

**restricción de unicidad.** Regla que determina que dos valores de una clave primaria o de una clave de un índice de unicidad no pueden ser iguales. También se denomina *limitación de unicidad*.

**restricciones de referencia.** Regla de integridad referencial que determina que los valores no nulos de la clave foránea sólo son válidos si también aparecen como valores de una clave padre.

**resumen (outline).** En el Kit de iniciación de OLAP, estructura que define todos los elementos de una base de datos dentro del Kit de iniciación de OLAP. Por ejemplo, un resumen contiene definiciones de dimensiones, miembros y fórmulas.

**retrotracción.** Proceso de restaurar datos modificados por sentencias de SQL al estado que tenían en el último punto de confirmación. Véase también “punto de coherencia” en la página 1597.

**revocar.** Eliminar un privilegio o una autorización de un identificador de autorización.

**RLF.** Véase “recurso de límite de recursos” en la página 1601.

**ROWID.** Véase “identificador de fila” en la página 1570.

**RR.** Véase “lectura repetible” en la página 1576.

**RRE (residual recovery entry).** Véase “entradas de recuperación residuales” en la página 1558.

**RRSAF (Recoverable Resource Manager Services Attachment Facility).** Véase “Recoverable Resource Manager Services” en la página 1599.

**RS.** Véase “estabilidad de lectura” en la página 1560.

**RUOW.** Véase “unidad de trabajo remota” en la página 1621.

**rutina.** Método definido por el usuario, función definida por el usuario o procedimiento almacenado.

**rutina de salida.** Programa que recibe el control de otro programa para realizar funciones específicas.

**rutina de SQL.** En DB2 Universal Database para OS/390, función definida por el usuario o procedimiento almacenado que está basado en código escrito en SQL.

**rutina externa.** Función, método o procedimiento escrito en un lenguaje principal y que posiblemente contiene sentencias de SQL.

## S

**salida de usuario.** Programa utilizado para interactuar con dispositivos de almacenamiento que no están directamente soportados por el sistema operativo. Cuando se invoca el programa de salida de usuario, el gestor de bases de datos pasa el control al archivo ejecutable. Sólo se puede invocar un programa de salida de usuario en una instancia del gestor de bases de datos.

**salto.** En APPN, parte de una ruta que no tiene nodos intermedios. Un salto consta de un grupo de transmisión individual que conecta nodos adyacentes.

**satélite.** Servidor DB2 que sincroniza con su grupo en el servidor de control de DB2 (“servidor de control de DB2” en la página 1608).

**SBCS.** Véase “juego de caracteres de un solo byte” en la página 1575.

**SCA (shared communications area).** En DB2 Universal Database para OS/390, área de comunicaciones compartida.

**SDK.** Véase “Software Developer’s Kit” en la página 1610.

**SDWA (system diagnostic work area).** En un entorno OS/390, el área de trabajo de diagnósticos del sistema.

**sección.** En DB2 Universal Database, segmento de un plan o paquete que contiene las estructuras ejecutables para una sentencia de SQL individual. Para la mayoría de sentencias de SQL, existe una sección en el plan para cada sentencia de SQL del programa fuente. Sin embargo, para las sentencias relacionadas con el cursor, las sentencias DECLARE, OPEN, FETCH y CLOSE hacen referencia a la misma sección porque cada una hace referencia a la sentencia SELECT que se menciona en la sentencia DECLARE CURSOR. Las sentencias de SQL como por ejemplo COMMIT, ROLLBACK y algunas sentencias SET no utilizan ninguna sección.

**segmentación del plan.** En DB2 Universal Database para OS/390, división de cada plan en secciones. Cuando es necesaria una sección, se lleva por separado a la agrupación de EDM.

**segmento.** Grupo de páginas que contienen una fila de una sola tabla. Véase también “espacio de tablas segmentado” en la página 1560.

**seguimiento de auditoría.** Datos, en forma de vía de acceso lógica que enlaza una secuencia de sucesos, que se utilizan para rastrear las transacciones que han afectado al contenido de un registro.

**seguridad de conversación.** En APPC, proceso que permite la validación de un identificador de usuario o identificador de grupo y una contraseña antes de establecer una conexión.

**seguridad de sesión.** En LU 6.2, verificación de la LU asociada y cifrado de datos de sesión. Función de SNA (Systems Network Architecture) que permite transmitir datos en forma cifrada.

**selección completa.** Subselección, cláusula de valores o varios de estos dos elementos que están combinados mediante operadores de conjunto. La selección completa especifica una tabla resultante. Si no se utiliza UNION, el resultado de la selección completa es el resultado de la subselección especificada.

**selección completa de inicialización.** Primera selección completa de una expresión de tabla común recurrente que obtiene los hijos directos del valor inicial a partir de la tabla fuente.

**selección completa escalar.** Selección completa que devuelve un valor individual: una fila de datos que consta exactamente de una sola columna.

**selección de función.** Véase “resolución de función” en la página 1603.

**sensibilidad de fetch.** (1) Especificación para las filas que ya se han captado. Este FETCH sólo tiene visibilidad en los cambios efectuados por el mismo cursor (es decir, los cambios efectuados utilizando UPDATE WHERE CURRENT OF o DELETE WHERE CURRENT OF que especifican el nombre del cursor). (2) Especificación de que este FETCH tiene visibilidad en todos los cambios efectuados por este cursor, así como en los cambios efectuados por otros cursores u otros procesos de aplicación. Esto hace que se capten siempre las filas de la tabla base de la sentencia SELECT del cursor.

## Glosario

**sensibilidad del cursor.** Especifica que el cursor tiene sensibilidad a los cambios efectuados en la base de datos después de que se materialice la tabla resultante. El cursor siempre es sensible a las actualizaciones y supresiones efectuadas utilizando el cursor (es decir, actualizaciones y supresiones posicionadas utilizando el mismo cursor). Cuando el valor actual de una fila ya no satisface la sentencia-select o el nombre-sentencia, dicha fila deja de estar visible mediante el cursor. Cuando una fila de la tabla resultante se suprime de la tabla base subyacente, la fila deja de estar visible mediante el cursor. La sensibilidad del cursor a los cambios realizados fuera de este cursor depende del hecho de que se utilice la sentencia SENSITIVE FETCH o la sentencia INSENSITIVE FETCH.

**sentencia.** Instrucción en un programa o procedimiento.

**sentencia de SQL compuesto.** Bloque de sentencias de SQL que se ejecutan en una sola llamada al servidor de aplicaciones.

**sentencia de SQL preparada.** En SQL, objeto con nombre que es la forma ejecutable de una sentencia de SQL que ha sido procesada por la sentencia PREPARE.

**sentencia ejecutable.** Sentencia de SQL que se puede incorporar en un programa de aplicación, preparar y ejecutar dinámicamente o emitir interactivamente.

**sentencia explicable.** Sentencia de SQL para la que se puede efectuar la operación de explicación. Las sentencias explicables son SELECT, UPDATE, INSERT, DELETE y VALUES.

**sentencia explicada.** Sentencia de SQL para la que se ha efectuado una operación de explicación.

**sentencias de SQL desencadenadas.** Conjunto de sentencias de SQL que se ejecuta cuando se activa un desencadenante y la condición de acción desencadenada se evalúa como verdadera. Las sentencias de SQL desencadenadas también se denominan *cuerno del desencadenante*.

**serie.** (1) En lenguajes de programación, formato de datos que se utiliza para almacenar y manipular texto. (2) Secuencia de bytes que puede representar caracteres.

**serie binaria.** En DB2 Universal Database, secuencia de bytes que no está asociada a un CCSID. Por ejemplo, el tipo de datos BLOB es una serie binaria. Véase también “identificador de juego de caracteres codificado” en la página 1570.

**serie corta.** (1) Serie de longitud fija o variable cuya longitud máxima es menor o igual que 254 bytes. (2) En DB2 Universal Database para OS/390, serie cuya longitud real, o serie de longitud variable cuya longitud máxima, es 255 bytes (o 127 caracteres de doble byte) o menos. Independientemente de la longitud, una serie LOB no es una serie corta. Véase también “serie larga” en la página 1607.

**serie de caracteres.** Secuencia de bytes que representa datos de bits, caracteres de un solo byte o una combinación de caracteres de un solo byte y de doble byte.

**serie de caracteres de contabilidad.** Información de contabilidad definida por el usuario que DB2 Connect envía a los servidores DRDA. Esta información se puede especificar desde la estación de trabajo cliente utilizando la API SQLESACT o la variable de entorno DB2ACCOUNT o desde la estación de trabajo DB2 Connect utilizando el parámetro de configuración DFT\_ACCOUNT\_STR del gestor de bases de datos.

**serie de caracteres mixtos.** Serie que contiene una mezcla de caracteres de un solo byte y de múltiples bytes. Sinónimo de *serie de datos mixtos*.

**serie de longitud fija.** Serie gráfica o de caracteres cuya longitud está especificada y no se puede modificar. Véase también “serie de longitud variable”.

**serie de longitud variable.** Serie de caracteres, gráfica o binaria cuya longitud no es fija pero puede oscilar entre unos límites establecidos. También se conoce como *serie de longitud que varía*.

**serie de sentencia.** En una sentencia de SQL dinámico, dentro de un entorno DB2 Universal Database para OS/390, el formato de serie de caracteres de la sentencia.

**serie gráfica.** Secuencia de caracteres DBCS.

**serie larga.** (1) Serie de longitud variable cuya longitud máxima supera los 254 bytes. (2) En DB2 Universal Database para OS/390, serie cuya longitud real, o serie de longitud variable cuya longitud máxima, es mayor que 255 bytes ó 127 caracteres de doble byte. Se considera que es una serie larga cualquier columna LOB, variable LOB de lenguaje principal o expresión cuya evaluación da un LOB como resultado. Véase también “serie corta” en la página 1606.

**servicios ampliados entre sistemas (cross-system extended services - XES).** Conjunto de servicios de OS/390 que permiten que varias instancias de una aplicación o subsistema, que se ejecutan en sistemas diferentes de un entorno Parallel Sysplex, lleven a cabo un compartimiento de datos de alto rendimiento y disponibilidad utilizando un recurso de acoplamiento.

**servicios de directorio.** Parte de los protocolos APPN que mantiene información acerca de la ubicación de los recursos de una red APPN.

**Servicios de directorio de red (Network Directory Services - NDS).** Base de datos global, duplicada y distribuida, de NetWare que mantiene información acerca de los recursos de la red y proporciona acceso a ellos. La base de datos del Directorio de NetWare dispone los objetos, cualesquiera que sea su ubicación física, en una estructura jerárquica en árbol llamada árbol de directorios.

**servicios del método de acceso.** Recurso que se utiliza para definir y reproducir archivos VSAM ordenados por clave.

**servicios de red.** Servicios existentes dentro de las unidades direccionables de red que controlan el funcionamiento de la red mediante sesiones SSCP-SSCP, SSCP-PU, SSCP-LU y CP-CP.

**servicios de topología y direccionamiento (topology and routing services, TRS).** Componente de un punto de control de APPN que gestiona la base de datos de topología y calcula rutas.

**servidor.** (1) En una red, nodo que proporciona recursos a otras estaciones; por ejemplo, un servidor de archivos, un servidor de impresoras, un servidor de correo. (2) (2) En un sistema de bases de datos federado, unidad de información que identifica una fuente de datos para un servidor federado. Esta información puede incluir el nombre del servidor, su tipo, su versión y el nombre del reiniciador que el servidor federado utiliza para comunicarse con la fuente de datos y recuperar datos de ella. (3) Destino de una petición de un petionario remoto. En el entorno DB2, la función de servidor la proporciona el recurso de datos distribuidos, que se utiliza para acceder a los datos de DB2 desde aplicaciones remotas. Véase también “servidor de aplicaciones”.

**servidor de aplicaciones.** En DB2 Universal Database para OS/390, destino de una petición de una aplicación remota. En el entorno DB2, la función de servidor de aplicaciones la proporciona el recurso de datos distribuidos y se utiliza para acceder a los datos de DB2 desde aplicaciones remotas. Véase también “petionario de aplicaciones” en la página 1592.

## Glosario

**servidor de archivos.** Estación de trabajo donde se ejecuta el software del sistema operativo NetWare y actúa como un servidor de red. DB2 utiliza el servidor de archivos para almacenar direcciones de servidores DB2, que los clientes DB2 recuperan para establecer una conexión cliente-servidor de IPX/SPX.

**servidor de bases de datos.** Destino de una petición de una aplicación local o un servidor de bases de datos intermedio. En el entorno DB2, la función de servidor de bases de datos la proporciona el recurso de datos distribuidos para acceder a los datos de DB2 desde aplicaciones locales o un servidor de bases de datos remotas que actúa como servidor de bases de datos intermediario.

**servidor de bases de datos intermedio.** Destino de una petición de un peticionario de aplicaciones local o de aplicaciones remoto que se está reenviando a otro servidor de bases de datos porque el objeto no existe en el servidor de bases de datos de destino. La petición remota se envía de forma transparente a otro servidor de bases de datos si el objeto al que hace referencia el nombre de tres partes no hace referencia a la ubicación local. Véase también “servidor de bases de datos”.

**servidor de control.** En el proceso de duplicación de datos de DB2, ubicación en la base de datos de las correspondientes definiciones de suscripción y de las tablas de control del programa Apply.

**servidor de control de DB2.** Sistema DB2 Universal Database que contiene la base de datos de control de satélites, SATCTLDB.

**servidor de nodos de la red.** Nodo de red APPN que proporciona servicios de red para sus unidades lógicas locales y nodos finales adyacentes.

**servidor de nombres de dominio (domain name server, DNS).** Servidor de red TCP/IP que gestiona un directorio distribuido el cual sirve para correlacionar nombres de sistema principal TCP/IP con direcciones IP.

**servidor destino.** En el proceso de duplicación de datos de DB2, ubicación en la base de datos de la tabla destino. Normalmente es también la ubicación del programa Apply.

**servidor Domino Go Web.** Servidor Web ofrecido por Lotus Corp. e IBM, que ofrece conexiones estables y seguras. ICAPI y GWAPI son las interfaces proporcionadas con este servidor.

**servidor federado.** Servidor de un sistema federado en el que están instalado DB2 EE o EEE, Relational Connect y el software cliente para las fuentes de datos a las que se accederá.

**servidor fuente.** En el proceso de duplicación de datos de DB2, ubicación en la base de datos de la fuente de duplicación y del programa Capture.

**sesión.** Conexión lógica entre dos estaciones o unidades direccionables de red (NAU) SNA que permite que las dos estaciones o unidades NAU se comuniquen.

**sesión de desconexión (UNBIND).** Petición para desactivar una sesión entre dos unidades lógicas (LU).

**sesión de supervisión.** Acción de supervisar un gestor de bases de datos o de reproducir información de un gestor de bases de datos supervisado anteriormente. El Supervisor de rendimiento de DB2 se utiliza para crear una sesión de supervisión y para seleccionar los objetos de la base de datos que se deben supervisar.

**sesión paralela.** En SNA, dos o más sesiones activas simultáneamente entre las dos mismas unidades lógicas. Cada sesión puede tener parámetros de sesión diferentes. Véase “sesión”.

**seudónimo.** Nombre alternativo que se utiliza para identificar una tabla, una vista, una base de datos o un apodo. Se puede utilizar un seudónimo en sentencias de SQL para hacer referencia a una tabla o vista que reside en el mismo subsistema DB2 o en un subsistema DB2 remoto.

**signatura de función.** Concatenación lógica de un nombre de función completamente calificado con los tipos de datos de todos sus parámetros. Cada función de un esquema debe tener una firma exclusiva.

**símbolo de acceso.** En DB2 Data Links Manager, clave cifrada asignada por el gestor de bases de datos que debe generarse para acceder a un archivo bajo el control de Data Links Manager.

**símbolo de coherencia.** En DB2 Universal Database, indicación de fecha y hora que se utiliza para generar el identificador exclusivo (identificador de versión en OS/390) para una aplicación.

**símbolo delimitador.** Constante de tipo serie, identificador delimitado, símbolo de operador o cualquiera de los caracteres especiales mostrados en los diagramas de sintaxis.

**símbolo de recuperación.** En DB2 Universal Database para OS/390, identificador de un elemento que se utiliza en la recuperación (por ejemplo *NID* o *URID*).

**símbolo (léxico).** Unidad sintáctica básica de un lenguaje de programación. Un símbolo (léxico) consta de uno o más caracteres, excluidos el carácter de espacio en blanco y los caracteres de una constante de tipo serie o identificador delimitado.

**símbolo ordinario.** Constante numérica, identificador común, identificador de sistema principal o palabra clave.

**simultaneidad.** Uso compartido y simultáneo de recursos por varios procesos de aplicación o usuarios interactivos.

**sin barrera.** Tipo de función definida por el usuario o de procedimiento almacenado que se define para ejecutarse en el proceso del gestor de bases de datos. No existe ninguna protección para el gestor de bases de datos frente a las modificaciones realizadas por esta función. Véase también “con barrera” en la página 1541.

**síncrono.** Perteneciente a dos o más procesos que dependen de las apariciones de sucesos específicos, por ejemplo una señal de temporización común. Véase también “asíncrono” en la página 1531.

**sinónimo.** En DB2 Universal Database para OS/390, nombre alternativo utilizado en SQL para una tabla o vista.

**sistema de bases de datos federado.** Un sistema DB2 federado es un tipo especial de sistema de gestión de bases de datos distribuidas (DBMS). Un sistema federado permite consultar y recuperar datos ubicados en otros gestores de bases de datos, por ejemplo Oracle, Sybase y Microsoft SQL Server. Las sentencias de SQL pueden hacer referencia a múltiples gestores de bases de datos o a bases de datos individuales en una sola sentencia. Por ejemplo, puede unir datos ubicados en una tabla de DB2 Universal Database, una tabla de Oracle y una vista de Sybase.

**Sistema de Control de la Información del Cliente (Customer Information Control System - CICS).** Programa bajo licencia de IBM que proporciona servicios de proceso de transacciones en línea y gestión para aplicaciones de gestión críticas. En la información de DB2 Universal Database para OS/390, este término representa el Servidor de transacciones CICS para OS/390, CICS/ESA y CICS/MVS.



## Glosario

**sistema de gestión de bases de datos (database management system - DBMS).** Véase “gestor de bases de datos” en la página 1565.

**sistema de gestión de bases de datos relacionales (relational database management system - RDBMS).** Conjunto de hardware y software que organiza y proporciona el acceso a una base de datos relacional.

**sistema de información geográfica (geographic information system - GIS).** Complejo de objetos, datos y aplicaciones que permite generar y analizar información espacial acerca de las características geográficas, incluidos los objetos que componen la superficie de la tierra (por ejemplo: ríos, bosques, colinas, desiertos) y objetos que la ocupan (por ejemplo: ciudades, residencias, edificios de oficinas, señales).

**sistema de nombres de dominio.** Sistema de bases de datos distribuidas utilizado por TCP/IP para correlacionar nombres de máquina legibles por el hombre con direcciones IP.

**sistema principal.** En TCP/IP, cualquier sistema que tenga como mínimo una dirección Internet asociada a él.

**sitio de agente.** En el Centro de depósito de datos, ubicación, definida por un solo nombre de sistema principal de red, donde se instala una aplicación agente de depósito.

**SMF (system management facility).** Véase “recurso de gestión del sistema” en la página 1600.

**SMS (storage management subsystem).** Véase “Subsistema de gestión de almacenamiento” en la página 1612.

**SNA.** Véase “Arquitectura de red de sistemas” en la página 1531.

**socket.** Interfaz de programación TCP/IP llamable utilizada por aplicaciones de red TCP/IP para comunicarse con asociados TCP/IP remotos.

**Software Developer’s Kit (SDK).** Producto de desarrollo de aplicaciones que permite desarrollar aplicaciones en una estación de trabajo cliente para tener acceso a servidores de bases de datos remotos incluidas bases de datos relacionales de sistema principal por medio de los productos DB2 Connect.

**sólo lectura.** Un archivo que tiene acceso de sólo lectura se puede leer pero no se puede actualizar o suprimir.

**soporte de base de datos federada.** Posibilidad de DB2 Universal Database que permite a los usuarios tener acceso de lectura a todos los gestores de bases de datos DB2 Universal Database y bases de datos Oracle con una sola sentencia de SQL.

**SPUFI (SQL Processor Using File Input).** Véase “Procesador de SQL que utiliza entrada de archivos” en la página 1611.

**SQL.** Véase “Lenguaje de Consulta Estructurada” en la página 1576.

**SQLCA.** Véase “área de comunicaciones de SQL” en la página 1530.

**SQLDA.** Véase “área de descriptores de SQL” en la página 1530.



**SQL dinámico.** Sentencias de SQL que se preparan y ejecutan en tiempo de ejecución. En SQL dinámico, la sentencia de SQL está contenida como una serie de caracteres en una variable del lenguaje principal y no es precompila. Véase también “SQL incorporado” y “SQL estático”.

**SQL estático.** Sentencias de SQL que están incorporadas dentro de un programa y se preparan durante el proceso de preparación del programa antes de que éste se ejecute. Una vez preparadas, las sentencias de SQL estático no cambian, aunque los valores de las variables de lenguaje principal especificadas por la sentencia pueden cambiar. Véase también “SQL incorporado” y “SQL dinámico”.

**SQL incorporado.** Sentencias de SQL codificadas dentro de un programa de aplicación. Véase “SQL estático”.

**SQL incorporado diferido.** En DB2 Universal Database, sentencias de SQL que no son completamente estáticas ni completamente dinámicas. Al igual que las sentencias estáticas, están incorporadas a una aplicación, pero como las sentencias dinámicas, se preparan durante la ejecución de la aplicación.

**SQL interactivo.** Serie de sentencias entradas por el usuario mediante una interfaz como el centro de mandatos o el procesador de la línea de mandatos. Estas sentencias se procesan como sentencias de SQL dinámicas. Por ejemplo, una sentencia SELECT interactiva puede procesarse dinámicamente utilizando las sentencias DECLARE CURSOR, PREPARE, DESCRIBE, OPEN, FETCH y CLOSE.

**SQL Processor Using File Input (Procesador de SQL que utiliza entrada de archivos - SPUFI).** (1) En DB2 Universal Database para OS/390, SPUFI formatea y visualiza los conjuntos de datos de salida utilizando el programa Interactive System Facility Browse. (2) En DB2 Universal Database para OS/390, recurso del subcomponente de conexión TSO que permite al usuario de DB2I ejecutar sentencias de SQL sin incorporarlas en un programa de aplicación.

**SSCP.** Véase “punto de control de servicios del sistema” en la página 1597.

**SSI (subsystem interface).** En un entorno OS/390, interfaz de subsistema.

**SSM (subsystem member).** En DB2 Universal Database para OS/390, miembro de subsistema.

**subagente.** Tipo de agente que actúa sobre subpeticiones. Una aplicación individual puede efectuar muchas peticiones y cada petición puede descomponerse en muchas subpeticiones. Por tanto, pueden existir varios subagentes trabajando para una misma aplicación. Todos los subagentes que trabajan para la aplicación están coordinados por el agente de coordinación de dicha aplicación. Véase también “agente de coordinación” en la página 1526.

**subcomponente.** Grupo de módulos de DB2 Universal Database para OS/390 estrechamente relacionados que trabajan conjuntamente para proporcionar una función general.

**subconsulta.** Sentencia SELECT dentro de la cláusula WHERE o HAVING de otra sentencia de SQL; sentencia de SQL anidada.

**subconsulta correlacionada.** Subconsulta que contiene una referencia correlacionada a una columna de una tabla que está fuera de la subconsulta.

**subconsulta de autorreferencia.** Subselección o selección completa dentro de una sentencia DELETE, INSERT o UPDATE que hace referencia a la misma tabla que es el objeto de la sentencia de SQL.

## Glosario

**subpágina.** En DB2 Universal Database para OS/390, unidad en la que se puede dividir una página física de índice.

**subsección de coordinador.** Subsección de una aplicación que inicia otras subsecciones (si existen) y devuelve resultados a la aplicación.

**subselección.** Forma de consulta que no incluye una cláusula ORDER BY, una cláusula UPDATE ni operadores UNION.

**subsistema.** En DB2 Universal Database para OS/390, instancia diferenciada de un sistema de bases de datos relacionales (RDBMS).

**Subsistema de gestión de almacenamiento (Storage Management Subsystem - SMS).** En un entorno OS/390, la finalidad de SMS es automatizar tanto como sea posible la gestión del almacenamiento físico, centralizando el control, automatizando tareas y proporcionando controles interactivos a los administradores del sistema. SMS puede reducir la preocupación de los usuarios respecto a los detalles físicos del rendimiento, del espacio y de la gestión de dispositivos.

**subsistema local.** RDBMS exclusivo al que está conectado directamente el usuario o un programa de aplicación (en el caso de DB2 Universal Database para OS/390, mediante uno de los recursos de conexión de DB2 Universal Database para OS/390).

**subsistema remoto.** En DB2 Universal Database para OS/390, cualquier RDBMS, excepto los *subsistemas locales*, con el que el usuario o la aplicación puede comunicarse. No es necesario que el subsistema sea remoto en un sentido físico e incluso puede funcionar en el mismo procesador bajo el mismo sistema OS/390.

**suceso desencadenante.** Suceso que hace que se active un desencadenante. En general, un suceso desencadenante es la inserción, supresión o actualización de filas en una tabla específica.

**suceso desencadenante.** En una definición de desencadenante, operación INSERT, DELETE o UPDATE que hará que se active el desencadenante.

**supervisor del sistema.** Colección de información relacionada con el estado del sistema de bases de datos a niveles de instancia, base de datos y aplicación. Esta información se almacena en elementos de datos, que se pueden examinar tomando instantáneas puntuales en el tiempo o utilizando el supervisor de sucesos para registrar cronológicamente la actividad del sistema durante un periodo de tiempo.

**supervisor del sistema de base de datos.** Conjunto de varias API de programación que supervisan información de rendimiento y de estado sobre el gestor de bases de datos, las bases de datos y las aplicaciones que utilizan el gestor de bases de datos y el DB2 Connect.

**Supervisor de rendimiento.** Herramienta que permite a los administradores de bases de datos utilizar una interfaz gráfica para supervisar el rendimiento de un sistema DB2 con la finalidad de realizar ajustes. Se puede acceder a esta herramienta desde el Centro de control.

**supervisor de sucesos.** Objeto de base de datos para supervisar y recoger datos sobre las actividades de la base de datos durante un periodo de tiempo. Por ejemplo, iniciar la base de datos puede ser un suceso que haga que el supervisor de sucesos realice un seguimiento del número de usuarios del sistema, tomando cada hora una instantánea de los ID de autorización que utilizan la base de datos.

**supervisor en línea.** Véase "Supervisor de rendimiento".

**supresión en cascada.** Forma en que DB2 Universal Database aplica restricciones de referencia al suprimir todas las filas descendentes de una fila padre ("fila padre" en la página 1563) suprimida.

**supresión múltiple.** Supresión de todas las filas de una tabla.

**suscripción.** Véase "conjunto de suscripción" en la página 1544.

**suscripción de duplicación.** Especificación para copiar datos modificados desde fuentes de duplicación a tablas destino a una hora y frecuencia especificadas, con la opción de ampliación de datos. Define toda la información que el programa Apply necesita para copiar datos.

**SYS1.LOGREC.** En un entorno OS/390, ayuda de servicio que contiene información importante sobre errores de programa y de hardware.

**Sysplex.** Véase "Parallel Sysplex" en la página 1589.

## T

**tabla.** Objeto de datos con nombre que consta de un número específico de columnas y algunas filas sin ordenar. Véase también "tabla base", "tabla temporal declarada" en la página 1617 y "tabla temporal" en la página 1617.

**tabla auxiliar.** Tabla que almacena columnas fuera de la tabla que en la que están definidas. Véase también "tabla base".

**tabla base.** (1) Tabla creada con la sentencia CREATE TABLE. Esta tabla tiene almacenados físicamente su descripción y sus datos en la base de datos. (2) En DB2 Universal Database para OS/390, tabla creada con la sentencia CREATE TABLE que contiene una definición de columna LOB. Los datos reales de columna LOB no se almacenan con esta tabla base. La tabla base contiene un identificador de fila para cada fila y una columna indicadora para cada una de sus columnas LOB. Véase también "tabla temporal declarada" en la página 1617, "tabla auxiliar", "vista" en la página 1625, "tabla resultante" en la página 1616 y "tabla temporal" en la página 1617.

**tabla base de agregación.** En el proceso de duplicación de datos de DB2, tipo de tabla destino que contiene datos recogidos a intervalos de una tabla fuente o tabla puntual en el tiempo.

**tabla CCD.** Véase "tabla de datos de cambios coherentes" en la página 1614.

**tabla CCD completa.** Tabla de datos de cambios coherentes que contiene todas las filas que satisfacen la vista fuente y predicados de la tabla o vista fuente. Véase también "tabla CCD no completa" en la página 1614.

**tabla CCD condensada.** En la duplicación de DB2, tabla de datos de cambios coherentes que sólo contiene el valor más actual para una fila. Este tipo de tabla es útil para transferir cambios a ubicaciones remotas y resumir actualizaciones in situ. Véase también "tabla CCD no condensada" en la página 1614.

**tabla CCD externa.** En duplicación de DB2, tabla de datos de cambios coherentes a la que es posible suscribirse directamente porque es una fuente de duplicación registrada. Posee su propia fila en la tabla de registro, donde aparece como SOURCE\_OWNER y SOURCE\_TABLE. Véase "tabla de datos de cambios coherentes" en la página 1614. Véase también "tabla CCD interna" en la página 1614.

## Glosario

**tabla CCD interna.** Tabla de datos de cambios coherentes a la que no es posible suscribirse directamente. Carece de una fila propia en la tabla de registro; aparece referenciada como CCD\_OWNER y CCD\_TABLE en la fila correspondiente a la fuente de duplicación asociada. Véase también “tabla CCD externa” en la página 1613.

**tabla CCD no completa.** En duplicación de DB2, tabla de datos de cambios coherentes que está vacía cuando se crea y a la que se añaden filas a medida que se realizan cambios en la fuente. Véase también “tabla CCD completa” en la página 1613.

**tabla CCD no condensada.** En duplicación de DB2, tabla de datos de cambios coherentes que contiene el histórico de los cambios efectuados en los valores de una fila. Este tipo de tabla es útil para fines de auditoría. Véase “tabla de datos de cambios coherentes”. Véase también “tabla CCD condensada” en la página 1613.

**tabla CD.** Véase “tabla de datos de cambio”.

**tabla con tipo.** Una tabla puede tener el tipo de datos de cada columna definido por separado o tener los tipos de las columnas basados en los atributos de un tipo estructurado definido por el usuario.

**tabla de agregación de cambios.** En el proceso de duplicación de datos de DB2, tipo de tabla destino que contiene agregaciones de datos basadas en cambios registrados para una tabla fuente.

**tabla de autorreferencia.** Tabla que es al mismo tiempo tabla padre y tabla dependiente en la misma restricción referencial.

**tabla de catálogo.** Cualquier tabla del catálogo de DB2 Universal Database creada por DB2 para cada base de datos. Por ejemplo, estas tablas contienen información acerca de las definiciones de objetos de base de datos, por ejemplo índices, vistas y tablas de usuario, así como información de seguridad acerca de la autorización que tienen los usuarios en dichos objetos. No se pueden crear o eliminar de forma explícita, pero se puede consultar y ver su contenido utilizando las vistas de catálogo.

**tabla de control.** En el proceso de duplicación de datos de DB2, tabla en la que se guardan definiciones de fuentes de duplicación y de suscripción u otra información de control de duplicación.

**tabla de control de recursos (resource control table, RCT).** En DB2 Universal Database para OS/390 con CICS, estructura del recurso de conexión de CICS, creada por parámetros de macro proporcionados por la ubicación, que define atributos de autorización y acceso para transacciones o grupos de transacciones.

**tabla de copia de usuario.** En el proceso de duplicación de datos de DB2, tabla de destino cuyo contenido coincide con la totalidad o parte de una tabla fuente y contiene sólo columnas de datos de usuario.

**tabla de datos de cambio.** Tabla de control de duplicación ubicada en el servidor fuente que contiene los datos cambiados de una tabla fuente de duplicación.

**tabla de datos de cambios coherentes (CCD).** En el proceso de duplicación de datos de DB2, tipo de tabla destino que se utiliza para auditar o transferir datos o ambas cosas. Véase también “tabla CCD completa” en la página 1613, “tabla CCD no completa”, “tabla CCD externa” en la página 1613, “tabla CCD interna” y “tabla CCD no condensada”.

**tabla de destino.** En el proceso de duplicación de datos de DB2, tabla situada en el servidor destino en la que se copian datos. Puede ser una tabla de copia de usuario, una tabla puntual en el tiempo, una tabla agregada base, una tabla agregada de cambios, una tabla de datos de cambios coherentes o una tabla de duplicado.

**tabla de dimensión.** Representación de una dimensión en un esquema en estrella. Cada fila de una tabla de dimensión representa todos los atributos de un miembro determinado de la dimensión. Véase también “dimensión” en la página 1554 y “esquema en estrella” en la página 1560.

**tabla de especificación de límite de recursos.** En DB2 Universal Database para OS/390, tabla definida localmente que especifica los límites que debe aplicar el servicio de límite de recursos.

**tabla de excepciones.** (1) Tabla creada por el usuario que refleja la definición de la tabla que se está cargando. (2) Tabla que contiene filas que vulneran las restricciones de referencia o las restricciones de comprobación de tabla que el programa de utilidad CHECK DATA encuentra.

**tabla de índice común.** Tabla de DB2 cuyas columnas de texto comparten un índice de texto común.

**tabla de la unidad de trabajo.** Tabla de control de la duplicación ubicada en el servidor fuente que contiene registros de confirmación leídos del diario o archivo de anotaciones de la base de datos. Los registros incluyen un identificador de unidad de recuperación que se puede utilizar para unir la tabla de unidad de trabajo y la tabla de datos de cambios para producir datos de cambios coherentes con la transacción. Para DB2, la tabla de unidad de trabajo incluye opcionalmente el identificador de correlación, que puede ser útil para realizar operaciones de auditoría.

**tabla de módulo de subsistema externa (ESMT).** En el entorno OS/390, nombre de la tabla de módulo de subsistema externa, que especifica qué módulos de conexión debe cargar el Sistema de gestión de información.

**tabla de parámetros de ajuste.** Tabla del servidor fuente que contiene información de sincronización utilizada por el programa Capture. La información incluye el tiempo que deben conservarse las filas en la tabla de datos de cambios, el tiempo que debe transcurrir para que los cambios se almacenen en una anotación cronológica o un registro por diario de la base de datos y la frecuencia con la que deben confirmarse los datos cambiados en las tablas de unidad de trabajo.

**tabla dependiente.** Tabla que depende de una restricción de referencia como mínimo.

**tabla de registro de anotaciones.** Tabla creada por Text Extender que contiene información acerca de los documentos de texto que se deben indexar.

**tabla de resumen.** Tabla cuya definición se basa en el resultado de una consulta y cuyos datos están en forma de resultados precalculados tomados de la tabla o tablas en las que se basa su definición.

**tabla de resumen automática (AST).** Tabla de resumen definida de tal forma que los cambios efectuados en las tablas subyacentes se colocan en cascada en la tabla de resumen inmediatamente y sin necesidad de una sentencia REFRESH TABLE. Véase “tabla de resumen”.

**tabla descendiente.** Tabla dependiente de otra tabla o que es descendiente de una tabla dependiente.

**tabla desencadenante.** Tabla para la cual se crea un desencadenante. Cuando el suceso desencadenante definido se produce en esta tabla, se activa el desencadenante.

## Glosario

**tabla de soporte administrativo.** Tabla utilizada por un expansor de DB2 para procesar peticiones de usuario sobre objetos de imagen, audio y vídeo. Algunas tablas de soporte administrativo identifican tablas y columnas de usuario que están habilitadas para un expansor. Otras tablas de soporte administrativo contienen información de atributos sobre objetos existentes en columnas habilitadas. También se denomina *tabla de metadatos*.

**tabla destino de duplicado.** Tabla de duplicación situada en el servidor destino que es un tipo de tabla destino con actualización en cualquier lugar.

**tabla de transición.** Tabla temporal que contiene todas las filas afectadas de la tabla sujeto en el estado que tenían antes o después de que se produjese el suceso desencadenante. Las sentencias de SQL desencadenadas en la definición de desencadenante pueden hacer referencia a la tabla de filas cambiadas en el estado anterior o el estado nuevo.

**tabla de usuario.** En el proceso de duplicación de datos de DB2, tabla creada para una aplicación y utilizada por ella antes de definirse como fuente de duplicación. Se utiliza como fuente para actualizaciones en tablas destino de sólo lectura, tablas de datos de cambios coherentes, duplicados y tablas de duplicado de filas.

**tabla factual.** (1) En el Kit de iniciación de OLAP, tabla o, en muchos casos, conjunto de tablas de DB2 que contiene todos los valores de datos para un cubo relacional. (2) Se compone de una o más tablas relacionales que contienen hechos, por ejemplo unidades vendidas o coste de las mercancías, y claves foráneas que enlazan la tabla factual con cada tabla de dimensión.

**tabla fuente.** En el proceso de duplicación de datos de DB2, tabla que contiene los datos que se deben copiar en una tabla destino. La tabla fuente puede ser una tabla fuente de duplicación, una tabla de datos de cambios o una tabla de datos de cambios coherentes. Véase también “tabla de destino” en la página 1615.

**tabla intermedia.** En duplicación de DB2, tabla de datos de cambios coherentes que se puede utilizar como fuente para actualizar datos en múltiples tablas de destino. Véase “tabla de datos de cambios coherentes” en la página 1614.

**tabla padre.** Tabla que es padre en una restricción referencial como mínimo.

**tabla puntual en el tiempo.** En el proceso de duplicación de datos de DB2, tipo de tabla destino cuyo contenido coincide total o parcialmente con una tabla fuente, con una columna del sistema añadida que identifica la hora aproximada en que se ha insertado o actualizado la fila determinada en el sistema fuente.

**tabla resultante.** Conjunto de filas producidas por la evaluación de una sentencia SELECT. Véase también “tabla temporal” en la página 1617.

**tablas disponibles continuamente (CAT).** Mecanismo para mantener en sincronismo dos copias de una tabla, donde cada copia está en una base de datos independiente. Esto hará de forma efectiva que la tabla esté continuamente disponible para las consultas y las actualizaciones. Durante una interrupción del suministro eléctrico donde una base de datos no esté disponible, los cambios se capturarán en la base de datos superviviente y, a continuación, se aplicarán a la base de datos que estaba inactiva cuando ésta esté disponible otra vez.

**tabla sujeto.** Tabla para la cual se crea un desencadenante. Cuando el suceso desencadenante definido se produce en esta tabla, se activa el desencadenante.

**tabla temporal.** (1) Tabla creada durante el proceso de una sentencia de SQL para que contenga resultados intermedios. Existen dos tipos de tablas temporales: tablas temporales de sistema de almacenamiento de espacios de tablas del sistema y tablas temporales declaradas de almacenamientos de tablas temporales de usuario. (2) Tabla que contiene datos temporales. Por ejemplo, las tablas temporales son útiles para mantener o clasificar resultados intermedios de consultas que contienen un gran número de filas. Las dos clases de tablas temporales, que se crean mediante sentencias de SQL diferentes, son la tabla temporal creada y la tabla temporal declarada. Véase también “tabla resultante” en la página 1616, “tabla temporal creada” y “tabla temporal declarada”.

**tabla temporal creada.** En DB2 Universal Database para OS/390, tabla que contiene datos temporales y se define con la sentencia de SQL CREATE GLOBAL TEMPORARY TABLE. La información acerca de las tablas temporales creadas se almacena en el catálogo de DB2, de modo que esta clase de tabla es permanente y se puede compartir entre procesos de partición. Véase “tabla temporal”. Véase también “tabla temporal declarada”.

**tabla temporal declarada.** Tabla que contiene datos temporales y se define con la sentencia de SQL DECLARE GLOBAL TEMPORARY TABLE. La información acerca de las tablas temporales declaradas no se almacena en el catálogo DB2, de modo que esta clase de tabla no es permanente y sólo la puede utilizar el proceso de aplicación que ha emitido la sentencia DECLARE. Véase también “tabla base” en la página 1613, “tabla temporal creada” y “tabla temporal”.

**tabla unida.** Tabla resultante intermedia que es el resultado de una “unión interna” en la página 1622 o una “unión externa” en la página 1622.

**tamaño del bloqueo.** Cantidad de datos controlados por un bloqueo de DB2 Universal Database para OS/390 en los datos de tabla; el valor puede ser una fila, una página, un LOB, una partición, una tabla o un espacio de tablas.

**tarea.** En el Centro de tareas, unidad de trabajo y sus acciones de planificación y tarea asociadas. Las tareas pueden establecerse para que se ejecuten según una planificación y pueden realizar diversas acciones basándose en el éxito o el fracaso de la tarea. Los scripts de DB2, los scripts de operación y los pasos de depósito son ejemplos de tareas. Véase también “acción de tarea” en la página 1525 y “paso” en la página 1590.

**tarea de origen.** En DB2 Universal Database para OS/390, agente primario de un grupo paralelo que recibe datos de otras unidades de ejecución (conocidas como *tareas paralelas*) que están ejecutando en paralelo partes de la consulta.

**tarea paralela.** En un entorno OS/390, unidad de ejecución que se crea dinámicamente para procesar una consulta en paralelo. Se implementa mediante un bloque de petición de servicio de MVS.

**TCB.** Véase “bloque de control de tareas” en la página 1533.

**TCP/IP.** Véase “Transmission Control Protocol/Internet Protocol” en la página 1620.

**temporización a petición.** Método para controlar la temporización de la duplicación para sistemas conectados ocasionalmente. Este método requiere que el usuario utilice el programa ASNSAT para operar los programas Capture y Apply. Véase también “temporización de sucesos” en la página 1618 y “temporización de intervalo”.

**temporización de intervalo.** En el proceso de duplicación de datos de DB2, el método más simple para controlar cuándo iniciar un ciclo de suscripción. El usuario debe especificar una fecha y una hora en que



## Glosario

debe comenzar un ciclo de suscripción, y establecer un intervalo de tiempo que describe la frecuencia con que se desea ejecutar el ciclo de suscripción. Véase también “temporización de sucesos” y “temporización a petición” en la página 1617.

**temporización de sucesos.** En el proceso de duplicación de datos de DB2, el método más preciso para controlar cuándo iniciar un ciclo de suscripción. Requiere que el usuario especifique un suceso y la hora en que desea que se procese el suceso. Véase también “temporización de intervalo” en la página 1617 y “temporización a petición” en la página 1617.

**terminación anómala.** Véase “terminación anómala de tarea”.

**terminación anómala.** (1) Error del sistema o acción del operador que hace que un trabajo termine de forma no satisfactoria. (2) En DB2, salida que no está bajo el control del programa, por ejemplo una trampa o segv.

**terminación anómala de tarea (abend).** Terminación de una tarea, un trabajo o un subsistema debido a una condición de error que los recursos de recuperación no pueden resolver durante la ejecución.

**terminador NUL.** En el lenguaje C, el valor que indica el final de una serie. Para series de caracteres, el terminador NUL es X'00'.

**territorio.** Porción del entorno local POSIX que se correlaciona con el código de país para su proceso interno por el gestor de bases de datos.

**texto de SQL optimizado.** Texto de SQL, producido por el recurso Explain, que está basado en la consulta utilizada realmente por el optimizador para elegir el plan de acceso. Esta consulta ha sido complementada y reescrita por los diversos componentes del compilador de SQL durante la compilación de sentencias. El texto se reconstruye a partir de su representación interna y difiere del texto de SQL original. La sentencia optimizada produce el mismo resultado que la sentencia original.

**tiempo de espera excedido.** Terminación anormal del subsistema DB2 Universal Database para OS/390 o de una aplicación debido a la no disponibilidad de los recursos. Se establecen especificaciones de instalación para determinar la cantidad de tiempo que DB2 Universal Database para OS/390 esperará recibir servicios de IRLM después de iniciarse y la cantidad de tiempo que IRLM esperará si un recurso solicitado por una aplicación no está disponible. Si se excede cualquiera de estas dos especificaciones de tiempo, se declara un tiempo de espera excedido.

**timeron.** Unidad de medida que se utiliza para hacer un cálculo relativo aproximado de los recursos, o coste, que el servidor de bases de datos necesita para ejecutar dos planes para la misma consulta. Los recursos calculados en la estimación incluyen costes ponderados de procesador y de E/S.

**tipificación estricta.** Proceso que garantiza que sólo las funciones definidas por el usuario y las operaciones que están definidas en un tipo diferenciado puedan aplicarse a dicho tipo. Por ejemplo, no se pueden comparar directamente dos tipos de moneda, tales como dólares canadienses y dólares americanos. Pero se puede utilizar una función definida por el usuario para convertir una moneda en la otra y luego realizar la comparación.

**tipo de datos.** En SQL, atributo de columnas, literales, variables del lenguaje principal, registros especiales y de los resultados de funciones y expresiones.

**tipo de datos definido por el usuario.** Véase “tipo diferenciado” en la página 1619.



**tipo de datos parametrizados.** Tipo de datos que se puede definir con una longitud, escala o precisión específica. Los datos de tipo serie y de tipo decimal están parametrizados.

**tipo definido por el usuario (UDT).** Tipo de datos que no es nativo respecto al gestor de bases de datos y ha sido creado por un usuario. En DB2 Universal Database, se utiliza el término tipo diferenciado en lugar de tipo definido por el usuario.

**tipo de LU.** Clasificación de una unidad lógica en términos del subconjunto específico de protocolos y opciones SNA que soporta para una sesión determinada. Específicamente, los valores permitidos en la petición de activación de sesión, la utilización de controles de corriente de datos, las cabeceras de gestión de función, los parámetros de unidad de petición, los valores de datos de detección y los protocolos de servicios de presentación tales como los asociados con las cabeceras de gestión de función.

**tipo de objeto.** (1) Número de 2 bytes que clasifica un objeto en el archivo de base de seguridad de un servidor de archivos NetWare. Por ejemplo, 062B representa el tipo de objeto de servidor de bases de datos DB2. (2) Categoría o agrupación de instancias de objetos que comparten comportamientos y características similares.

**tipo de PU.** En SNA, clasificación de una unidad física (PU) de acuerdo con el tipo de nodo en el que reside.

**tipo diferenciado.** Tipo de datos definido por el usuario que está representado internamente como un tipo existente (su tipo fuente), pero se considera que es un tipo diferente e incompatible para fines semánticos.

**tipo diferenciado definido por el usuario.** Véase “tipo diferenciado”.

**tipo fuente.** Tipo existente que se utiliza para representar internamente un tipo diferenciado.

**Tivoli Space Manager.** Característica del producto Tivoli Storage Manager que maneja la entrada y salida de archivos en un soporte de almacenamiento secundario basándose en los accesos a archivo reales del sistema de archivos nativo primario. Esta característica funciona con Data Links Manager para permitir que los archivos DATALINK se almacenen en un sistema de archivos de tamaño virtualmente infinito.

**Tivoli Storage Manager (TSM).** Producto cliente/servidor que proporciona servicios de gestión de almacenamiento y de acceso a datos en un entorno heterogéneo. TSM da soporte a varios métodos de comunicación, proporciona recursos administrativos para gestionar la copia y almacenamiento de archivos, y recursos para planificar operaciones de copia.

**TM de IMS.** Gestor de transacciones del sistema de gestión de información.

**TMP (terminal monitor program).** En un entorno OS/390, programa supervisor de terminales. Este programa se conecta al procesador de mandatos DSN proporcionado por DB2, el cual, a su vez, se conecta al programa de aplicación.

**TP.** Véase “programa de transacciones” en la página 1595.

**transacción.** (1) Intercambio de información entre una estación de trabajo y un programa, dos estaciones de trabajo o entre dos programas que realizan una acción o logran un resultado determinado. Un ejemplo de transacción es la entrada del ingreso de un cliente y la actualización del saldo del cliente. Sinónimo de “unidad de trabajo” en la página 1621. (2) Invocación individual de Net.Data. Si se utiliza Net.Data permanente, una transacción puede abarcar varias invocaciones de Net.Data.

## Glosario

**transacción conversacional.** En APPC, dos o más programas que se comunican utilizando los servicios de unidades lógicas (las LU).

**transacción distribuida.** Transacción que actualiza datos en más de una base de datos. Véase también “confirmación en dos fases” en la página 1543.

**transacción dudosa.** Transacción en la que finaliza satisfactoriamente una fase en una confirmación de dos fases, pero el sistema falla antes de que se pueda completar una fase subsiguiente.

**transacción global.** Unidad de trabajo en un entorno de proceso de transacciones distribuidas en el que son necesarios varios gestores de recursos.

**transacción no coordinada.** Transacción que accede a más de un recurso, pero cuya confirmación o retroacción no está siendo coordinada por un gestor de transacciones.

**transacción rechazada.** En el proceso de duplicación de datos de DB2, transacción que contiene una o más actualizaciones de tablas de duplicado que están anticuadas respecto a la tabla fuente.

**transformación.** En Centro de depósito de datos, operación realizada sobre datos. “Pivot” y “cleanse” son tipos de transformaciones.

**transformador.** Programa que actúa sobre datos de depósito. Centro de depósito de datos proporciona dos tipos de transformadores: transformadores estadísticos, que proporcionan estadísticas sobre los datos de una o más tablas, y transformadores de depósito, que preparan los datos para el análisis. Los transformadores tienen tipos de pasos correspondientes para los tipos de manipulación de datos que realizan los pasos; por ejemplo un paso de limpieza utiliza el transformador para limpiar datos (Clean Data).

**Transmission Control Protocol/Internet Protocol (TCP/IP).** Conjunto de protocolos de comunicaciones que proporciona funciones de conectividad de igual a igual para redes locales y de área amplia.

**truncamiento.** Proceso de eliminar parte del resultado de una operación cuando excede la capacidad de almacenamiento o memoria.

**truncamiento del archivo de anotaciones.** En DB2 Universal Database para OS/390, proceso mediante el cual se establece una dirección de byte relativa de inicio explícita. Esta RBA es el punto en el cual se escribirá el próximo byte de datos de registro cronológico.

**TSO (time-sharing option).** Véase “Opción de compartimiento de tiempo” en la página 1586.

## U

**ubicación.** Nombre exclusivo de un servidor de bases de datos. Una aplicación utiliza el nombre de ubicación para acceder a un servidor de bases de datos DB2. Véase también “nombre de ubicación” en la página 1585.

**UCS-2.** Juego de caracteres universal, codificado en 2 octetos, lo que significa que los caracteres se representan en 16 bits por carácter.

**UDF.** Véase “función definida por el usuario” en la página 1564.

**UDT.** Véase “tipo definido por el usuario” en la página 1619.

**Unicode.** Esquema internacional de codificación de caracteres que es un subconjunto del estándar ISO 10646. Cada carácter soportado se define utilizando un código exclusivo de 2 bytes. Véase también “ASCII” en la página 1531 y “EBCDIC” en la página 1556.

**unidad de recuperación .** Secuencia recuperable de operaciones dentro de un gestor de recursos individual, como por ejemplo una instancia de DB2 Universal Database para OS/390. Véase también “unidad de trabajo”.

**unidad de trabajo.** Secuencia recuperable de operaciones dentro de un proceso de aplicación. En todo momento, un proceso de aplicación es una unidad de trabajo única, pero el ciclo de vida de un proceso de aplicación puede implicar muchas unidades de trabajo como resultado de operaciones de confirmación y retrotracción. En una operación de *actualización de múltiples ubicaciones* de DB2 Universal Database para OS/390, una unidad de trabajo individual puede incluir varias *unidades de recuperación*. Sinónimo de “transacción” en la página 1619. Véase también “unidad de recuperación” y “actualización de múltiples ubicaciones” en la página 1525.

**unidad de trabajo distribuida (DUOW).** Unidad de trabajo que permite someter sentencias de SQL a varios sistemas de gestión de bases de datos relacionales, pero a razón de un sólo sistema por cada sentencia de SQL.

**unidad de trabajo remota (RUOW).** Una unidad de trabajo remota permite a un usuario o a un programa de aplicación leer o actualizar datos en una ubicación por unidad de trabajo. Soporta el acceso a una base de datos dentro de una unidad de trabajo. Mientras que un programa de aplicación puede actualizar varias bases de datos remotas, sólo puede acceder a una base de datos de una unidad de trabajo. Véase “unidad de trabajo”.

**unidad direccionable de red (NAU).** Fuente o destino de la información transmitida por la red de control de vías de acceso. Una NAU puede ser una unidad lógica (LU), una unidad física (PU), un punto de control (CP) o un punto de control de servicios del sistema (SSCP). Véase también “nombre de red” en la página 1584.

**unidad física (PU).** Componente que gestiona y supervisa los recursos (tales como, enlaces conectados y estaciones de enlace adyacentes) asociados a un nodo, de acuerdo con lo solicitado por un SSCP a través de una sesión SSCP-PU. Un SSCP activa una sesión con la PU a fin de gestionar indirectamente, por medio de la PU, los recursos del nodo, tales como enlaces conectados. Este término es aplicable sólo a los nodos de tipo 2.0, 4 y 5. Véase también “punto de control” en la página 1597.

**unidad lógica dependiente (DLU).** Unidad lógica que necesita la ayuda de un punto de control de servicios del sistema (SSCP) para crear una instancia de una sesión de LU-LU. Véase “unidad lógica independiente”.

**unidad lógica de trabajo (LUW).** Proceso que un programa ejecuta entre puntos de sincronización.

**unidad lógica independiente (ILU).** Unidad lógica que es capaz de activar una sesión de LU-LU sin la ayuda de un punto de control de servicios del sistema (SSCP). Una ILU no tiene una sesión de SSCP-LU. Véase también “unidad lógica dependiente”.

**unidad lógica 6.2 (LU 6.2).** Tipo de LU que soporta sesiones entre dos aplicaciones que utilizan APPC.

**unidad lógica (LU).** (1) En SNA, puerto a través del cual un usuario final accede a la red SNA para comunicarse con otro usuario final. Una unidad lógica puede dar soporte a muchas sesiones con otras

## Glosario

unidades lógicas. (2) En un entorno OS/390, punto de acceso mediante el cual un programa de aplicación accede a la red SNA para comunicarse con otro programa de aplicación. Véase también “nombre de LU” en la página 1584.

**unidad lógica (LU) asociada.** (1) En SNA, participante remoto de una sesión. (2) Punto de acceso de la red SNA que está conectado al subsistema local DB2 Universal Database para OS/390 mediante una conversación VTAM.

**unión.** Operación relacional de SQL que permite recuperar datos de dos o más tablas basándose en valores de columna coincidentes. Véase también “unión externa derecha”, “unión externa izquierda”, “unión externa” y “unión interna”.

**unión compatible con la partición.** Unión en la que todas las filas unidas residen en la misma partición de base de datos. Véase “unión”.

**unión de difusión.** Unión en la que todas las particiones de una tabla se envían a todas las particiones de la base de datos.

**unión de equivalencia.** Operación de unión en la que la condición de unión tiene el formato *expresión = expresión*.

**unión dirigida.** Operación relacional en la que todas las filas de una o ambas tablas unidas se recalculan aleatoriamente y se dirigen hacia nuevas particiones de base de datos basándose en el predicado de unión. Si todas las columnas clave de particionamiento de una tabla participan en los predicados de unión de equivalencia, se recalcula aleatoriamente la otra tabla; de lo contrario, (si existe al menos un predicado de unión de equivalencia), se recalculan aleatoriamente ambas tablas. Véase “unión”.

**unión externa.** (1) Método de unión en el que una columna que no es común a todas las tablas que se están uniendo se convierte en parte de la tabla resultante. (2) Resultado de una operación de unión que incluye las filas coincidentes de las dos tablas que se están uniendo y conserva algunas o todas las filas no coincidentes de las tablas que se están uniendo. Véase “unión”. Véase también “unión interna”, “unión externa completa”, “unión externa izquierda” y “unión externa derecha”.

**unión externa completa.** Resultado de una operación de unión de SQL que incluye las filas seleccionadas de las dos tablas que se unen y conserva las filas no seleccionadas de ambas tablas. Véase “unión”. Véase también “unión externa”, “unión externa izquierda” y “unión externa derecha”.

**unión externa derecha.** En DB2 Universal Database, resultado de una operación de unión que incluye las filas coincidentes de las dos tablas que se unen y conserva las filas no coincidentes del segundo operando de la unión. Véase “unión”. Véase también “unión externa izquierda” y “unión externa completa”.

**unión externa izquierda.** En DB2 Universal Database, resultado de una operación de unión que incluye las filas coincidentes de las dos tablas que se unen y conserva las filas no coincidentes de la primera tabla. Véase “unión”. Véase también “unión externa derecha”.

**unión interna.** Método de unión en el que se eliminan de la tabla resultante las columnas que no sean comunes a todas las tablas que se están uniendo. Véase “unión”. Véase también “unión externa”.

**unión ordenada.** Resultado de la unión de dos tablas cuando dichas tablas residen en un grupo de particiones de base de datos de una sola partición en la misma partición de la base de datos o cuando

están en el mismo grupo de particiones de base de datos, tienen el mismo número de columnas de particionamiento, las columnas con compatibles con la partición, ambas tablas utilizan la misma función de particionamiento y las parejas de las columnas clave de particionamiento correspondientes participan en los predicados de unión de equivalencia (equijoin).

**Extended UNIX Code (EUC).** Protocolo que puede dar soporte a juegos de caracteres de 1 a 4 bytes de longitud. EUC es un medio de especificar un conjunto de páginas de códigos en lugar de ser un esquema de codificación de páginas de códigos propiamente dicho. Es la alternativa de UNIX a los esquemas de codificación de páginas de códigos de doble byte (DBCS) de PC.

**UR.** Véase “lectura no confirmada (UR)” en la página 1576.

**UR de cancelación anómala diferida.** En DB2 Universal Database para OS/390, unidad de recuperación (UR), que estaba en vuelo o en cancelación anómala, que se ha interrumpido por una anomalía o cancelación del sistema y no ha completado la retrotracción durante el reinicio.

**URE (unit of recovery element).** En DB2 Universal Database para OS/390, unidad de elemento de recuperación.

**URID (ID de unidad de recuperación).** En DB2 Universal Database para OS/390, LOGRBA del primer registro de anotaciones de una unidad de recuperación. El URID también aparece en todos los registros de anotaciones subsiguientes de esa unidad de recuperación.

**URL.** Véase “localizador uniforme de recursos” en la página 1578.

**UT.** En DB2 Universal Database para OS/390, acceso sólo de utilidad.

**UTC.** Véase “Hora universal coordinada” en la página 1568.

**UTF-16.** Unicode Transformation Format, formato de codificación de 16 bits, que está diseñado para proporcionar valores de código para más de un millón de caracteres y un superconjunto de UCS-2. El valor de CCSID para los datos en formato UTF-16 es 1200. DB2 para OS/390 y z/OS soporta UTF-16 en campos de datos gráficos.

**UTF-8.** Unicode Transformation Format, formato de codificación de 8 bits, que está diseñado para facilitar el uso con los sistemas basados en ASCII existentes. El valor de CCSID para los datos en formato UTF-8 es 1208. DB2 para OS/390 y z/OS soporta UTF-8 en campos de datos mixtos.

## V

**valor.** (1) Contenido alfabético o numérico de un campo o una variable. (2) Unidad más pequeña de datos manipulada en SQL. (3) Elemento de datos específico en la intersección de una columna y una fila.

**valor de distribución de columnas.** Estadísticas que describen los valores más frecuentes de alguna columna o los valores cuantiles. Estos valores se utilizan en el optimizador para ayudar a determinar el mejor plan de acceso.

**valor de indicación de fecha y hora.** Valor cuyo tipo de datos es DATE, TIME o TIMESTAMP.

**valor nulo.** Posición de parámetro para la que no se ha especificado ningún valor.

## Glosario

**variable.** Elemento de datos que especifica un valor que se puede cambiar. Véase también “constante” en la página 1544.

**variable de lenguaje principal.** En un programa de aplicación, variable a la que hacen referencia sentencias de SQL intercaladas. Las variables del lenguaje principal son variables de programación del programa de aplicación y son el mecanismo principal para pasar datos entre las tablas de la base de datos y las áreas de trabajo del programa de aplicación.

**variable del lenguaje principal con terminación NUL.** En DB2 Universal Database para OS/390, variable del lenguaje principal de longitud variable en la que el final de los datos se indica mediante la presencia de un terminador NUL.

**variable de referencia a archivo.** Variable del lenguaje principal utilizada para indicar que los datos residen en un archivo del cliente en lugar de residir en un almacenamiento intermedio del cliente.

**variable de rendimiento.** Estadística derivada de los datos de rendimiento obtenidos del gestor de bases de datos. La expresión para esta variable puede definirla el usuario.

**variable de rendimiento definida por el usuario.** Variable de rendimiento creada por un usuario y que se añade al perfil de variables de rendimiento.

**variable de transición.** Variable que sólo es válida en los desencadenantes FOR EACH ROW. Permite tener acceso a los valores de transición para la fila actual. Una variable de transición antigua es el valor de la fila antes de aplicar la modificación y la variable de transición nueva es el valor de la fila después de aplicar la modificación.

**variable indicadora.** Variable utilizada para representar el valor nulo en un programa de aplicación. Si el valor para la columna seleccionada es nulo, se coloca un valor negativo en la variable indicadora.

**variable localizadora.** Variable del lenguaje principal que contiene el localizador que representa un valor de LOB en el servidor de aplicaciones.

**versión.** En DB2 Universal Database para OS/390, miembro de un conjunto de programas, módulos DBRM, paquetes u objetos LOB similares. He aquí unos ejemplos:

- Una versión de un programa es el código fuente que resulta de precompilar el programa. La versión de un programa se identifica mediante el nombre del programa y una indicación de fecha y hora (símbolo de coherencia).
- Una versión de un DBRM es el DBRM que resulta de precompilar un programa. La versión de DBRM se identifica mediante el mismo nombre de programa e indicación de fecha y hora que en una versión de programa correspondiente.
- Una versión de un paquete es el resultado de enlazar un DBRM dentro de un sistema de bases de datos determinado. La versión de paquete se identifica mediante el mismo nombre de programa y símbolo de coherencia que para el DBRM.
- Una versión de un LOB es una copia de un valor LOB en un momento determinado. El número de versión de un LOB se guarda en la entrada de índice auxiliar correspondiente al LOB.

**vía de acceso.** Método que el gestor de bases de datos selecciona para recuperar datos de una tabla específica. Por ejemplo, una vía de acceso puede incluir la utilización de un índice, una exploración secuencial o una combinación de ambas.

**vía de acceso.** (1) En un sistema operativo, una vía de acceso es la ruta a un archivo específico a través de un sistema de archivos. (2) En un entorno de red, una vía de acceso es la ruta entre dos nodos cualesquiera. (3) Véase “vía de acceso de SQL”.

**vía de acceso absoluta.** Vía de acceso completa de un objeto. Los nombres de vía de acceso absoluta empiezan en el nivel más alto o directorio “raíz” (que se identifica por un carácter de barra inclinada (/) o barra inclinada invertida (\)).

**vía de acceso actual.** Lista ordenada de nombres de esquema utilizados en la resolución de referencias no calificadas hechas a funciones y tipos de datos. En el SQL dinámico, la vía de acceso de función actual se encuentra en el registro especial CURRENT PATH. En el SQL estático, la vía de acceso se define en la opción FUNCPATH de los mandatos PREP y BIND.

**vía de acceso de anotación cronológica dual.** Vía de acceso de anotación cronológica secundaria para mantener copias duplicadas de archivos archivados en línea en la anotación cronológica activa.

**vía de acceso de función.** Lista ordenada de nombres de esquema que restringe el ámbito de búsqueda para invocaciones de funciones no calificadas y proporciona un árbitro final para el proceso de selección de función.

**vía de acceso de SQL.** En DB2 Universal Database para OS/390, lista ordenada de nombres de esquema que se utiliza en la resolución de referencias no calificadas en funciones definidas por el usuario, tipos diferenciados y procedimientos almacenados. En el SQL dinámico, la vía de acceso actual se encuentra en el registro especial CURRENT PATH. En el SQL estático, la vía de acceso actual está definida en la opción de enlace PATH.

**vía de acceso de ubicación.** Subconjunto de la sintaxis abreviada de la vía de acceso de ubicación definida por XPath. Secuencia de identificadores de XML para identificar un elemento o atributo de XML. Se utiliza en las funciones de extracción definidas por el usuario para identificar el sujeto a extraer y se utiliza en las funciones de búsqueda definidas por el usuario de Text Extender para identificar los criterios de búsqueda.

**vigencia de los datos.** En DB2 Universal Database para OS/390, estado en el cual los datos recuperados de una variable de lenguaje principal del programa son una copia de los datos de la tabla base.

**Virtual Storage Access Method (VSAM).** Método de acceso para el proceso directo o secuencial de registros de longitud fija o variable en dispositivos de acceso directo. Los registros de un archivo VSAM pueden estar dispuestos en una secuencia lógica según un campo de clave (secuencia de clave), en la secuencia física con la que se escriben en el archivo (secuencia de entrada) o según el número relativo de registro.

**Virtual Telecommunications Access Method (VTAM).** En un entorno OS/390, programa bajo licencia de IBM que controla la transmisión y el flujo de datos en una red SNA.

**vista.** (1) Tabla lógica que consta de datos generados por una consulta. (2) Modo de examinar la información acerca de objetos o contenida en objetos. Cada vista puede revelar información diferente acerca de sus objetos. Véase también “tabla base” en la página 1613.

**vista con tipo.** Una vista puede tener el tipo de datos de cada columna obtenido de la tabla de resultados o tener los tipos de las columnas basados en los atributos de un tipo estructurado definido por el usuario.



## Glosario

**vista de catálogo.** (1) En DB2 Universal Database, vistas SYSCAT y SYSSTAT de la tabla de catálogo. (2) Vista que contiene información acerca de las tablas y columnas que Text Extender habilita para ser utilizadas. Asimismo vista de la tabla del sistema creada por Text Extender para la administración.

**vista de usuario.** En el modelo de datos lógico, modelo o representación de información crítica que necesita la empresa.

**vista no operativa.** (1) Vista que ya no se puede utilizar porque el privilegio SELECT en una tabla o vista de la es dependiente la vista se ha revocado desde el definidor de la vista. (2) Un objeto del que depende la definición de la vista se ha eliminado (o posiblemente ha quedado no operativo en el caso de otra vista).

**vista relacionada.** Vista que utiliza, o depende de, otro objeto, por ejemplo la vista padre o una tabla.

**vista subyacente.** En DB2 Universal Database para OS/390, vista sobre la cual está definida, directa o indirectamente, otra vista.

**Visual Explain.** Herramienta que permite a los administradores de bases de datos y a los programadores de aplicaciones utilizar una interfaz gráfica para visualizar y analizar información detallada acerca del plan de acceso de una sentencia de SQL determinada. Se puede acceder a las tareas proporcionadas por esta herramienta desde el Centro de control.

**VSAM.** Véase “Virtual Storage Access Method” en la página 1625.

**VTAM.** Véase “Virtual Telecommunications Access Method” en la página 1625.

**vuelta atrás.** Proceso que consiste en deshacer los cambios no confirmados que ha efectuado un proceso de aplicación. Una vuelta atrás puede ser necesaria en el caso de que se produzca una anomalía en la parte de un proceso de aplicación o como resultado de una situación de punto muerto.

## W

**WTO.** Véase “escribir a operador” en la página 1558.

**WTOR.** “Escribir a operador” (WTO) con respuesta.

## X

**XCF.** Véase “recurso de acoplamiento entre sistemas” en la página 1600.

**XES.** Véase “servicios ampliados entre sistemas” en la página 1607.

**XID.** ID de estación de intercambio.

**XML.** Véase “lenguaje de marcación extensible” en la página 1576.

**XRF.** Véase “recurso de recuperación ampliado” en la página 1601.



**Y**

**ya verificado.** Opción de seguridad de SNA LU 6.2 que permite a DB2 Universal Database para OS/390 proporcionar el identificador de autorización verificado del usuario al asignar una conversación. El usuario no es validado por el subsistema asociado.

**Z**

**z/OS.** Sistema operativo para la línea de productos eServer que soporta almacenamiento real de 64 bits.



---

## Apéndice R. Utilización de la biblioteca de DB2

La biblioteca de DB2 Universal Database consta de ayuda en línea, manuales (PDF y HTML) y programas de ejemplo en formato HTML. Esta sección describe la información proporcionada y cómo puede acceder a ella.

Para acceder "en línea" a información de productos, puede utilizar el Centro de información. Para obtener más información, consulte el apartado "Acceso a información mediante el Centro de información" en la página 1645. En la Web puede visualizar información sobre tareas, manuales de DB2, resolución de problemas, programas de ejemplo e información sobre DB2.

---

### Archivos PDF y manuales impresos sobre DB2

#### Información sobre DB2

La tabla siguiente clasifica los manuales de DB2 en cuatro categorías:

##### Información de guía y consulta sobre DB2

Estos manuales contienen información básica sobre DB2 para todas las plataformas.

##### Información de instalación y configuración sobre DB2

Estos manuales están pensados para un sistema DB2 que se utiliza en una plataforma determinada. Por ejemplo, existen manuales de *Guía rápida de iniciación* diferentes para DB2 sobre OS/2, Windows y plataformas basadas en UNIX.

##### Programas de ejemplo en HTML para varias plataformas

Estos ejemplos son la versión HTML de los programas de ejemplo que se instalan con el Application Development Client. Están pensados para fines informativos y no sustituyen a los programas propiamente dichos.

##### Notas del release

Estos archivos contienen información de última hora que no se pudo incluir en los manuales de DB2.

Los manuales de instalación, las notas del release y las guías de aprendizaje se pueden visualizar directamente en formato HTML desde el CD-ROM del producto. La mayoría de los manuales pueden visualizarse en formato HTML desde el CD-ROM del producto y pueden visualizarse e imprimirse en formato PDF (Adobe Acrobat) desde el CD-ROM de publicaciones de DB2.

Puede también solicitar un ejemplar impreso a IBM; vea “Solicitud de los manuales impresos” en la página 1640. La tabla siguiente lista los manuales que se pueden solicitar.

En las plataformas OS/2 y Windows, puede instalar los archivos HTML en el directorio `sqllib\doc\html`. La información sobre DB2 está traducida a varios idiomas, pero no toda la información está disponible en todos los idiomas. Cuando la información no está disponible en un idioma determinado, se proporciona en el idioma inglés.

En las plataformas UNIX, puede instalar los archivos HTML en varios idiomas, en los directorios `doc/%L/html`, donde `%L` representa el entorno nacional. Para obtener más información, consulte el manual *Guía rápida de iniciación*.

Puede obtener manuales de DB2 y acceder a la información de varias maneras:

- “Visualización de información en línea” en la página 1644
- “Búsqueda de información en línea” en la página 1648
- “Solicitud de los manuales impresos” en la página 1640
- “Impresión de los manuales PDF” en la página 1640

Tabla 150. Información sobre DB2

| Nombre                                          | Descripción                                                                                                                                                                                                                                                               | Número de documento                 | Directorio de HTML |
|-------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------|--------------------|
|                                                 |                                                                                                                                                                                                                                                                           | Nombre de archivo PDF               |                    |
| <b>Información de guía y consulta sobre DB2</b> |                                                                                                                                                                                                                                                                           |                                     |                    |
| <i>Administration Guide</i>                     | <p><i>Administration Guide: Planning</i> proporciona una visión general de conceptos sobre bases de datos, información sobre cuestiones de diseño (tal como el diseño lógico y físico de una base de datos) y una exposición sobre el tema de la alta disponibilidad.</p> | SC09-2946<br>db2d1x70               | db2d0              |
|                                                 | <p><i>Administration Guide: Implementation</i> proporciona información sobre cuestiones de implantación, tales como la implantación del diseño de base de datos, el acceso a bases de datos, la auditoría, la copia y recuperación.</p>                                   | SC09-2944<br>db2d2x70               |                    |
|                                                 | <p><i>Administration Guide: Performance</i> proporciona información sobre el entorno de base de datos y la evaluación y ajuste del rendimiento de aplicaciones.</p>                                                                                                       | SC09-2945<br>db2d3x70               |                    |
|                                                 | <p>En Norteamérica, puede solicitar los tres volúmenes del manual <i>Administration Guide</i>, en lengua inglesa, utilizando el número de documento SBOF-8934.</p>                                                                                                        |                                     |                    |
| <i>Administrative API Reference</i>             | <p>Describe las interfaces de programación de aplicaciones (las API) de DB2 y las estructuras de datos que puede utilizar para gestionar las bases de datos. Este manual también explica cómo invocar las API desde las aplicaciones.</p>                                 | SC09-2947<br>db2b0x70               | db2b0              |
| <i>Application Building Guide</i>               | <p>Proporciona información para configurar el entorno e instrucciones paso a paso para compilar, enlazar y ejecutar aplicaciones DB2 en Windows, OS/2 y plataformas basadas en UNIX.</p>                                                                                  | SC09-2948<br>db2axx70               | db2ax              |
| <i>APPC, CPI-C, and SNA Sense Codes</i>         | <p>Proporciona información general sobre APPC, CPI-C y los códigos de detección SNA que pueden aparecer al utilizar productos DB2 Universal Database.</p>                                                                                                                 | Sin número de documento<br>db2apx70 | db2ap              |
|                                                 | <p>Solo está disponible en formato HTML.</p>                                                                                                                                                                                                                              |                                     |                    |

Tabla 150. Información sobre DB2 (continuación)

| Nombre                                                     | Descripción                                                                                                                                                                                                                                                                                                                                                                                                                 | Número de documento     | Directorio de HTML |
|------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|--------------------|
|                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                             | Nombre de archivo PDF   |                    |
| <i>Application Development Guide</i>                       | Explica cómo desarrollar aplicaciones que acceden a bases de datos DB2 mediante SQL incorporado o Java (JDBC y SQLJ). Los temas tratados incluyen la escritura de procedimientos almacenados, la escritura de funciones definidas por el usuario, la creación de tipos definidos por el usuario, la utilización de desencadenantes y el desarrollo de aplicaciones en entornos particionados o mediante sistemas federados. | SC09-2949               | db2a0              |
|                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                             | db2a0x70                |                    |
| <i>CLI Guide and Reference</i>                             | Explica la forma de desarrollar aplicaciones que acceden a bases de datos DB2 a través de la Interfaz de Nivel de Llamada de DB2, que es una interfaz SQL invocable que es compatible con la especificación ODBC de Microsoft.                                                                                                                                                                                              | SC09-2950               | db2l0              |
|                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                             | db2l0x70                |                    |
| <i>Consulta de mandatos</i>                                | Explica cómo utilizar el procesador de línea de mandatos y describe los mandatos de DB2 que puede utilizar para gestionar la base de datos.                                                                                                                                                                                                                                                                                 | SC10-3495               | db2n0              |
|                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                             | db2n0x70                |                    |
| <i>Connectivity Supplement</i>                             | Proporciona información de configuración y consulta sobre cómo utilizar DB2 para AS/400, DB2 para OS/390, DB2 para MVS o DB2 para VM como peticionarios de aplicaciones DRDA con servidores DB2 Universal Database. Este manual también describe cómo utilizar servidores de aplicaciones DRDA con peticionarios de aplicaciones DB2 Connect.                                                                               | Sin número de documento | db2h1              |
|                                                            | Solo está disponible en los formatos HTML y PDF.                                                                                                                                                                                                                                                                                                                                                                            |                         |                    |
| <i>Data Movement Utilities Guide and Reference</i>         | Explica cómo utilizar los programas de utilidad de DB2, tales como import, export, load, AutoLoader y DPROP, los cuales facilitan el movimiento de los datos.                                                                                                                                                                                                                                                               | SC09-2955               | db2dm              |
|                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                             | db2dmx70                |                    |
| <i>Data Warehouse Center Administration Guide</i>          | Proporciona información sobre cómo crear y mantener un depósito de datos utilizando el Centro de depósito de datos.                                                                                                                                                                                                                                                                                                         | SC26-9993               | db2dd              |
|                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                             | db2ddx70                |                    |
| <i>Data Warehouse Center Application Integration Guide</i> | Proporciona información para ayudar a los programadores a integrar aplicaciones mediante el Centro de depósito de datos y el Gestor de catálogos de información.                                                                                                                                                                                                                                                            | SC26-9994               | db2ad              |
|                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                             | db2adx70                |                    |

Tabla 150. Información sobre DB2 (continuación)

| Nombre                                                                  | Descripción                                                                                                                                                                                                                                                                                | Número de documento                 | Directorio de HTML |
|-------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------|--------------------|
|                                                                         |                                                                                                                                                                                                                                                                                            | Nombre de archivo PDF               |                    |
| <i>DB2 Connect User's Guide</i>                                         | Proporciona conceptos, información sobre programación e información general de utilización sobre los productos DB2 Connect.                                                                                                                                                                | SC09-2954<br>db2c0x70               | db2c0              |
| <i>DB2 Query Patroller Administration Guide</i>                         | Proporciona una visión general sobre el funcionamiento del sistema Query Patroller de DB2, información específica de utilización y administración e información sobre tareas para los programas de utilidad administrativos de la interfaz gráfica de usuario.                             | SC09-2958<br>db2dwx70               | db2dw              |
| <i>DB2 Query Patroller User's Guide</i>                                 | Describe cómo utilizar las herramientas y funciones de DB2 Query Patroller.                                                                                                                                                                                                                | SC09-2960<br>db2wwx70               | db2ww              |
| <i>Glosario</i>                                                         | Proporciona definiciones de términos utilizados en DB2 y en sus componentes.<br><br>Está disponible en formato HTML y en la publicación <i>Consulta de SQL</i> .                                                                                                                           | Sin número de documento<br>db2t0x70 | db2t0              |
| <i>Image, Audio, and Video Extenders Administration and Programming</i> | Proporciona información general sobre los expansores de DB2, e información sobre la administración y configuración de los expansores de imagen, audio y vídeo, y su utilización en la programación. Incluye información de consulta, información de diagnóstico (con mensajes) y ejemplos. | SC26-9929<br>dmbu7x70               | dmbu7              |
| <i>Information Catalog Manager Administration Guide</i>                 | Proporciona información de guía para la gestión de catálogos de información.                                                                                                                                                                                                               | SC26-9995<br>db2dix70               | db2di              |
| <i>Information Catalog Manager Programming Guide and Reference</i>      | Proporciona definiciones para las interfaces con arquitectura del Gestor de catálogos de información.                                                                                                                                                                                      | SC26-9997<br>db2bix70               | db2bi              |
| <i>Information Catalog Manager User's Guide</i>                         | Proporciona información sobre la utilización de la interfaz de usuario del Gestor de catálogos de información.                                                                                                                                                                             | SC26-9996<br>db2aix70               | db2ai              |

Tabla 150. Información sobre DB2 (continuación)

| Nombre                                                           | Descripción                                                                                                                                                                                                                                                                                                                                                                               | Número de documento                                                              | Directorio de HTML |
|------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------|--------------------|
|                                                                  |                                                                                                                                                                                                                                                                                                                                                                                           | Nombre de archivo PDF                                                            |                    |
| <i>Suplemento de instalación y configuración</i>                 | Sirve de guía para planificar, instalar y configurar clientes DB2 específicos de una plataforma. Este suplemento contiene información sobre la creación de enlaces, la configuración de comunicaciones de cliente y servidor, herramientas de GUI para DB2, DRDA AS, la instalación distribuida, la configuración de peticiones distribuidas y el acceso a fuentes de datos heterogéneas. | GC10-3487<br>db2iyx70                                                            | db2iy              |
| <i>Consulta de mensajes</i>                                      | Contiene los mensajes y códigos que emite DB2, el Gestor de catálogos de información y el Centro de depósito de datos, y describe las acciones que el usuario debe emprender.<br><br>En Norteamérica, puede solicitar ambos volúmenes del manual Consulta de mensajes, en lengua inglesa, utilizando el número de documento SBOF-8932.                                                    | Volumen 1<br>GC10-3493<br><br>db2m1x70<br>Volumen 2<br>GC10-3494<br><br>db2m2x70 | db2m0              |
| <i>OLAP Integration Server Administration Guide</i>              | Explica cómo utilizar el componente Gestor de Administración del Servidor de Integración de OLAP.                                                                                                                                                                                                                                                                                         | SC27-0782<br>db2dpx70                                                            | n/d                |
| <i>OLAP Integration Server Metaoutline User's Guide</i>          | Explica cómo crear y llenar con datos metabocetos OLAP utilizando la interfaz estándar OLAP Metaoutline (no mediante el Metaoutline Assistant).                                                                                                                                                                                                                                           | SC27-0784<br>db2upx70                                                            | n/d                |
| <i>OLAP Integration Server Model User's Guide</i>                | Explica cómo crear modelos OLAP utilizando la Interfaz de Modelos de OLAP (no mediante el Asistente de Modelos).                                                                                                                                                                                                                                                                          | SC27-0783<br>db2lpx70                                                            | n/d                |
| <i>Guía del usuario y de configuración de OLAP</i>               | Proporciona información de configuración e instalación sobre el Kit de arranque de OLAP.                                                                                                                                                                                                                                                                                                  | SC10-3526<br>db2ipx70                                                            | db2ip              |
| <i>OLAP Spreadsheet Add-in Guía del usuario para Excel</i>       | Describe cómo utilizar el programa de hoja de cálculo Excel para analizar datos de OLAP.                                                                                                                                                                                                                                                                                                  | SC10-3550<br>db2epx70                                                            | db2ep              |
| <i>OLAP Spreadsheet Add-in Guía del usuario para Lotus 1-2-3</i> | Describe cómo utilizar el programa de hoja de cálculo Lotus 1-2-3 para analizar datos de OLAP.                                                                                                                                                                                                                                                                                            | SC10-3551<br>db2tpx70                                                            | db2tp              |



Tabla 150. Información sobre DB2 (continuación)

| Nombre                                                 | Descripción                                                                                                                                                                                                                                                                                                               | Número de documento   | Directorio de HTML |
|--------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|--------------------|
|                                                        |                                                                                                                                                                                                                                                                                                                           | Nombre de archivo PDF |                    |
| <i>Replication Guide and Reference</i>                 | Proporciona información sobre la planificación, configuración, administración y utilización de las herramientas de duplicación de IBM que se ofrecen con DB2.                                                                                                                                                             | SC26-9920             | db2e0              |
|                                                        |                                                                                                                                                                                                                                                                                                                           | db2e0x70              |                    |
| <i>Spatial Extender Guía del usuario y de consulta</i> | Proporciona información sobre la instalación, configuración, administración, programación y resolución de problemas para el Spatial Extender. También proporciona descripciones importantes sobre conceptos de datos espaciales y ofrece información de consulta (mensajes y SQL) que es específica del Spatial Extender. | SC10-3528             | db2sb              |
|                                                        |                                                                                                                                                                                                                                                                                                                           | db2sbx70              |                    |
| <i>Guía de iniciación de SQL</i>                       | Proporciona conceptos básicos sobre SQL y ofrece ejemplos de muchas estructuras sintácticas y tareas.                                                                                                                                                                                                                     | SC10-3496             | db2y0              |
|                                                        |                                                                                                                                                                                                                                                                                                                           | db2y0x70              |                    |
| <i>Consulta de SQL, Volumen 1 y Volumen 2</i>          | Describe la sintaxis, la semántica y las normas del lenguaje SQL. Este manual también incluye información sobre las incompatibilidades entre releases, los límites del producto y las vistas de catálogo.                                                                                                                 | Volumen 1             | db2s0              |
|                                                        |                                                                                                                                                                                                                                                                                                                           | SC10-3497             |                    |
|                                                        |                                                                                                                                                                                                                                                                                                                           | db2s1x70              |                    |
|                                                        | En Norteamérica, puede solicitar ambos volúmenes del manual <i>Consulta de SQL</i> , en lengua inglesa, utilizando el número de documento SBOF-8933.                                                                                                                                                                      | Volumen 2             |                    |
|                                                        |                                                                                                                                                                                                                                                                                                                           | SC10-3549             |                    |
|                                                        |                                                                                                                                                                                                                                                                                                                           | db2s2x70              |                    |
| <i>System Monitor Guide and Reference</i>              | Describe cómo recoger distintos tipos de información sobre bases de datos y el gestor de bases de datos. Este manual explica cómo utilizar la información para comprender la actividad de una base de datos, mejorar su rendimiento y determinar la causa de los problemas.                                               | SC09-2956             | db2f0              |
|                                                        |                                                                                                                                                                                                                                                                                                                           | db2f0x70              |                    |
| <i>Text Extender Administración y programación</i>     | Proporciona información general sobre los expansores de DB2, e información sobre la administración y configuración del expansor de texto y su utilización en la programación. Incluye información de consulta, información de diagnóstico (con mensajes) y ejemplos.                                                      | SC10-3527             | desu9              |
|                                                        |                                                                                                                                                                                                                                                                                                                           | desu9x70              |                    |

Tabla 150. Información sobre DB2 (continuación)

| Nombre                                                                                         | Descripción                                                                                                                                                                                                                                                                                                                       | Número de documento   | Directorio de HTML |
|------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|--------------------|
|                                                                                                |                                                                                                                                                                                                                                                                                                                                   | Nombre de archivo PDF |                    |
| <i>Troubleshooting Guide</i>                                                                   | Le ayuda a determinar la causa de los errores, realizar la recuperación para un problema y utilizar herramientas de diagnóstico en colaboración con el Servicio de Asistencia al Cliente de DB2.                                                                                                                                  | GC09-2850<br>db2p0x70 | db2p0              |
| <i>Novedades</i>                                                                               | Describe las nuevas características, funciones y mejoras de DB2 Universal Database, Versión 7.                                                                                                                                                                                                                                    | SC10-3498<br>db2q0x70 | db2q0              |
| <b>Información de instalación y configuración sobre DB2</b>                                    |                                                                                                                                                                                                                                                                                                                                   |                       |                    |
| <i>DB2 Connect Enterprise Edition para OS/2 y Windows Guía rápida de iniciación, Versión 7</i> | Proporciona información sobre la planificación, migración, instalación y configuración de DB2 Connect Enterprise Edition en los sistemas operativos OS/2 y Sistemas operativos Windows de 32 bits. Este manual también contiene información sobre la instalación y configuración de muchos clientes a los que se da soporte.      | GC10-3486<br>db2c6x70 | db2c6              |
| <i>DB2 Connect Enterprise Edition para UNIX Guía rápida de iniciación</i>                      | Ofrece información sobre la planificación, migración, instalación, configuración y realización de tareas para DB2 Connect Enterprise Edition en plataformas basadas en UNIX. Este manual también contiene información sobre la instalación y configuración de muchos clientes a los que se da soporte.                            | GC10-3485<br>db2cyx70 | db2cy              |
| <i>DB2 Connect Personal Edition Quick Beginnings</i>                                           | Proporciona información sobre la planificación, migración, instalación, configuración y realización de tareas para DB2 Connect Personal Edition en el OS/2 y Sistemas operativos Windows de 32 bits. Este manual también contiene información sobre la instalación y configuración de todos los clientes a los que se da soporte. | GC09-2967<br>db2c1x70 | db2c1              |
| <i>DB2 Connect Personal Edition Quick Beginnings for Linux</i>                                 | Proporciona información sobre la planificación, instalación, migración y configuración de DB2 Connect Personal Edition en todas las distribuciones Linux soportadas.                                                                                                                                                              | GC09-2962<br>db2c4x70 | db2c4              |
| <i>DB2 Data Links Manager Guía rápida de iniciación</i>                                        | Proporciona información sobre la planificación, instalación, configuración y realización de tareas en DB2 Data Links Manager para los sistemas operativos AIX y Windows de 32 bits.                                                                                                                                               | GC10-3488<br>db2z6x70 | db2z6              |

Tabla 150. Información sobre DB2 (continuación)

| Nombre                                                                | Descripción                                                                                                                                                                                                                                                                                | Número de documento   | Directorio de HTML |
|-----------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|--------------------|
|                                                                       |                                                                                                                                                                                                                                                                                            | Nombre de archivo PDF |                    |
| <i>DB2 Enterprise - Extended Edition for UNIX Quick Beginnings</i>    | Ofrece información sobre la planificación, instalación y configuración de DB2 Enterprise - Extended Edition en plataformas basadas en UNIX. Este manual también contiene información sobre la instalación y configuración de muchos clientes a los que se da soporte.                      | GC09-2964<br>db2v3x70 | db2v3              |
| <i>DB2 Enterprise - Extended Edition for Windows Quick Beginnings</i> | Proporciona información sobre la planificación, instalación, configuración de DB2 Enterprise - Extended Edition para los sistemas operativos Windows de 32 bits. Este manual también contiene información sobre la instalación y configuración de muchos clientes a los que se da soporte. | GC09-2963<br>db2v6x70 | db2v6              |
| <i>DB2 para OS/2 Guía rápida de iniciación</i>                        | Ofrece información sobre la planificación, instalación, migración y configuración de DB2 Universal Database en el sistema operativo OS/2. Este manual también contiene información sobre la instalación y configuración de muchos clientes a los que se da soporte.                        | GC10-3489<br>db2i2x70 | db2i2              |
| <i>DB2 para UNIX Guía rápida de iniciación</i>                        | Ofrece información sobre la planificación, instalación, migración y configuración de DB2 Universal Database en plataformas basadas en UNIX. Este manual también contiene información sobre la instalación y configuración de muchos clientes a los que se da soporte.                      | GC10-3491<br>db2ixx70 | db2ix              |
| <i>DB2 para Windows Guía rápida de iniciación</i>                     | Proporciona información sobre la planificación, instalación, migración y configuración de DB2 Universal Database en Sistemas operativos Windows de 32 bits. Este manual también contiene información sobre la instalación y configuración de muchos clientes a los que se da soporte.      | GC10-3492<br>db2i6x70 | db2i6              |
| <i>DB2 Personal Edition Guía rápida de iniciación</i>                 | Proporciona información sobre la planificación, instalación, migración y configuración de DB2 Universal Database Personal Edition en el OS/2 y Sistemas operativos Windows de 32 bits.                                                                                                     | GC10-3490<br>db2i1x70 | db2i1              |
| <i>DB2 Personal Edition Quick Beginnings for Linux</i>                | Proporciona información sobre la planificación, instalación, migración y configuración de DB2 Universal Database Personal Edition en todas las distribuciones Linux soportadas.                                                                                                            | GC09-2972<br>db2i4x70 | db2i4              |

Tabla 150. Información sobre DB2 (continuación)

| Nombre                                                      | Descripción                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Número de documento                        | Directorio de HTML |
|-------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------|--------------------|
|                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Nombre de archivo PDF                      |                    |
| <i>DB2 Query Patroller Installation Guide</i>               | Proporciona información sobre la instalación de DB2 Query Patroller.                                                                                                                                                                                                                                                                                                                                                                                                                                                              | GC09-2959<br>db2iwx70                      | db2iw              |
| <i>DB2 Warehouse Manager Guía de instalación</i>            | Proporciona información sobre la instalación de agentes de depósito, transformadores de depósito y el Gestor de catálogos de información.                                                                                                                                                                                                                                                                                                                                                                                         | GC10-3659<br>db2idx70                      | db2id              |
| <b>Programas de ejemplo en HTML para varias plataformas</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                            |                    |
| Programas de ejemplo en HTML                                | <p>Proporciona los programas de ejemplo en formato HTML para los lenguajes de programación de todas las plataformas soportadas por DB2. Los programas de ejemplo se ofrecen sólo con fines informativos. No todos los programas de ejemplo están disponibles en todos los lenguajes de programación. Los ejemplos en formato HTML sólo pueden utilizarse si está instalado DB2 Application Development Client.</p> <p>Para obtener más información sobre los programas, consulte el manual <i>Application Building Guide</i>.</p> | Sin número de documento                    | db2hs              |
| <b>Notas del release</b>                                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                            |                    |
| <i>Notas del release de DB2 Connect</i>                     | Proporciona información de última hora que no se pudo incluir en los manuales de DB2 Connect.                                                                                                                                                                                                                                                                                                                                                                                                                                     | Ver nota 2.                                | db2cr              |
| <i>Notas de instalación de DB2</i>                          | Proporciona información de última hora, específica de la instalación, que no se pudo incluir en los manuales de DB2.                                                                                                                                                                                                                                                                                                                                                                                                              | Sólo disponible en el CD-ROM del producto. |                    |
| <i>Notas del release de DB2</i>                             | Proporciona información de última hora, referente a todos los productos y características de DB2, que no se pudo incluir en los manuales de DB2.                                                                                                                                                                                                                                                                                                                                                                                  | Ver nota 2.                                | db2ir              |

**Notas:**

1. El carácter *x* que ocupa la sexta posición en el nombre de archivo indica el idioma en que está escrito el manual. Por ejemplo, el nombre de archivo db2d0e70 identifica la versión inglesa del manual *Administration Guide* y el nombre de archivo db2d0f70 identifica la versión francesa del mismo

manual. En la posición sexta de los nombres de archivo se utilizan las letras siguientes para indicar el idioma del manual:

| <b>Idioma</b>       | <b>Identificador</b> |
|---------------------|----------------------|
| Alemán              | g                    |
| Búlgaro             | u                    |
| Checo               | x                    |
| Chino simplificado  | c                    |
| Chino tradicional   | t                    |
| Coreano             | k                    |
| Danés               | d                    |
| Esloveno            | l                    |
| Español             | z                    |
| Finés               | y                    |
| Francés             | f                    |
| Griego              | a                    |
| Holandés            | q                    |
| Húngaro             | h                    |
| Inglés              | e                    |
| Italiano            | i                    |
| Japonés             | j                    |
| Noruego             | n                    |
| Polaco              | p                    |
| Portugués brasileño | b                    |
| Portugués           | v                    |
| Ruso                | r                    |
| Sueco               | s                    |
| Turco               | m                    |

2. La información de última hora que no se pudo incluir en los manuales de DB2 se encuentra en las Notas del release, en formato HTML y en forma de archivo ASCII. La versión en formato HTML puede consultarse desde el Centro de información y en los CD-ROM del producto. Para visualizar el archivo ASCII:
  - En las plataformas basadas en UNIX, vea el archivo Release.Notes. Este archivo está situado en el directorio DB2DIR/Readme/%L, donde %L representa el entorno nacional y DB2DIR representa:
    - /usr/lpp/db2\_07\_01 en AIX
    - /opt/IBMdb2/V7.1 en HP-UX, PTX, Solaris, y Silicon Graphics IRIX
    - /usr/IBMdb2/V7.1 en Linux.
  - En otras plataformas, vea el archivo RELEASE.TXT. Este archivo reside en el directorio donde está instalado el producto. En las plataformas OS/2, puede también hacer una doble pulsación sobre la carpeta **IBM DB2** y luego sobre el icono **Notas del release**.

## Impresión de los manuales PDF

Si prefiere tener copias impresas de los manuales, puede imprimir los archivos PDF contenidos en el CD-ROM de publicaciones de DB2. Mediante Adobe Acrobat Reader, puede imprimir el manual completo o un rango específico de páginas. Para conocer el nombre de archivo de cada manual de la biblioteca, vea la Tabla 150 en la página 1631.

Puede obtener la última versión de Adobe Acrobat Reader en el sitio Web de Adobe, que se encuentra en <http://www.adobe.com>.

Los archivos PDF contenidos en el CD-ROM de publicaciones de DB2 tienen PDF como extensión de archivo. Para acceder a los archivos PDF:

1. Inserte el CD-ROM de publicaciones de DB2. En las plataformas basadas en UNIX, monte el CD-ROM de publicaciones de DB2. Consulte el manual *Guía rápida de iniciación* para conocer los procedimientos de montaje del CD-ROM.
2. Arranque Acrobat Reader.
3. Abra el archivo PDF deseado que se encuentra en una de las ubicaciones siguientes:
  - En las plataformas OS/2 y Windows:  
el directorio `x:\doc\idioma`, donde `x` representa la unidad de CD-ROM e `idioma` representa el código de país de dos caracteres correspondiente al idioma del usuario (por ejemplo, EN para el inglés).
  - En las plataformas basadas en UNIX:  
el directorio `/cdrom/doc/%L` del CD-ROM, donde `/cdrom` representa el punto de montaje del CD-ROM y `%L` representa el entorno nacional deseado.

Puede también copiar los archivos PDF del CD-ROM a una unidad local o de red y leerlos desde allí.

## Solicitud de los manuales impresos

Puede solicitar los manuales impresos de DB2 en forma individual o como colección de manuales (en Norteamérica sólo), utilizando en este segundo caso un número de documento SBOF (sold bill of forms). Para solicitar manuales, consulte al concesionario o representante de ventas autorizado de IBM, o llame a los números 1-800-879-2755 (Estados Unidos) o 1-800-IBM-4YOU (Canadá). Puede también solicitar manuales desde la página Web de publicaciones, situada en <http://www.elink.ibm.com/pbl/pbl>.

Puede adquirir dos colecciones de manuales. SBOF-8935 proporciona información de consulta y de utilización sobre DB2 Warehouse Manager.

SBOF-8931 proporciona información de consulta y de utilización sobre todos los demás productos y características de DB2 Universal Database. La tabla siguiente lista el contenido de cada colección de manuales:

Tabla 151. Pedido de los manuales impresos

| Número SBOF | Manuales incluidos                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SBOF-8931   | <ul style="list-style-type: none"> <li>• Administration Guide: Planning</li> <li>• Administration Guide: Implementation</li> <li>• Administration Guide: Performance</li> <li>• Administrative API Reference</li> <li>• Application Building Guide</li> <li>• Application Development Guide</li> <li>• CLI Guide and Reference</li> <li>• Command Reference</li> <li>• Data Movement Utilities Guide and Reference</li> <li>• Data Warehouse Center Administration Guide</li> <li>• Data Warehouse Center Application Integration Guide</li> <li>• DB2 Connect User's Guide</li> <li>• Installation and Configuration Supplement</li> <li>• Image, Audio, and Video Extenders Administration and Programming</li> <li>• Message Reference, Volúmenes 1 y 2</li> </ul> | <ul style="list-style-type: none"> <li>• OLAP Integration Server Administration Guide</li> <li>• OLAP Integration Server Metaoutline User's Guide</li> <li>• OLAP Integration Server Model User's Guide</li> <li>• OLAP Integration Server User's Guide</li> <li>• OLAP Setup and User's Guide</li> <li>• OLAP Spreadsheet Add-in User's Guide for Excel</li> <li>• OLAP Spreadsheet Add-in User's Guide for Lotus 1-2-3</li> <li>• Replication Guide and Reference</li> <li>• Spatial Extender Administration and Programming Guide</li> <li>• SQL Getting Started</li> <li>• SQL Reference, Volúmenes 1 y 2</li> <li>• System Monitor Guide and Reference</li> <li>• Text Extender Administration and Programming</li> <li>• Troubleshooting Guide</li> <li>• What's New</li> </ul> |
| SBOF-8935   | <ul style="list-style-type: none"> <li>• Information Catalog Manager Administration Guide</li> <li>• Information Catalog Manager User's Guide</li> <li>• Information Catalog Manager Programming Guide and Reference</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | <ul style="list-style-type: none"> <li>• Query Patroller Administration Guide</li> <li>• Query Patroller User's Guide</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

---

## Documentación en línea de DB2

### Acceso a la ayuda en línea

Existe ayuda en línea para todos los componentes de DB2. La tabla siguiente describe los diversos tipos de ayuda.

| Tipo de ayuda                                               | Contenido                                                                                                                                                                                                                                       | Cómo acceder...                                                                                                                                                                                                                                                                                                                                                        |
|-------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Ayuda para mandatos</i>                                  | Explica la sintaxis de los mandatos del procesador de línea de mandatos.                                                                                                                                                                        | Desde el procesador de línea de mandatos en modalidad interactiva, especifique:<br><br><i>? mandato</i><br><br>donde <i>mandato</i> representa una palabra clave o el mandato completo.<br><br>Por ejemplo, <i>? catalog</i> visualiza ayuda para todos los mandatos CATALOG, mientras que <i>? catalog database</i> visualiza ayuda para el mandato CATALOG DATABASE. |
| <i>Ayuda para el Asistente de configuración del cliente</i> | Explica las tareas que el usuario puede realizar en una ventana o cuaderno. La ayuda incluye información general e información sobre los requisitos previos que debe conocer, y describe cómo utilizar los controles de una ventana o cuaderno. | Desde una ventana o cuaderno, pulse el botón <b>Ayuda</b> o pulse la tecla <b>F1</b> .                                                                                                                                                                                                                                                                                 |
| <i>Ayuda para el Centro de mandatos</i>                     |                                                                                                                                                                                                                                                 |                                                                                                                                                                                                                                                                                                                                                                        |
| <i>Ayuda para el Centro de control</i>                      |                                                                                                                                                                                                                                                 |                                                                                                                                                                                                                                                                                                                                                                        |
| <i>Ayuda para el Centro de depósito de datos</i>            |                                                                                                                                                                                                                                                 |                                                                                                                                                                                                                                                                                                                                                                        |
| <i>Ayuda para el Analizador de sucesos</i>                  |                                                                                                                                                                                                                                                 |                                                                                                                                                                                                                                                                                                                                                                        |
| <i>Ayuda para el Gestor de catálogos de información</i>     |                                                                                                                                                                                                                                                 |                                                                                                                                                                                                                                                                                                                                                                        |
| <i>Ayuda para el Centro de administración de satélites</i>  |                                                                                                                                                                                                                                                 |                                                                                                                                                                                                                                                                                                                                                                        |
| <i>Ayuda para el Centro de scripts</i>                      |                                                                                                                                                                                                                                                 |                                                                                                                                                                                                                                                                                                                                                                        |

---



| Tipo de ayuda              | Contenido                                                                 | Cómo acceder...                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|----------------------------|---------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Ayuda para mensajes</i> | Describe la causa de un mensaje y la acción que debe realizar el usuario. | <p>Desde el procesador de línea de mandatos en modalidad interactiva, especifique:</p> <pre data-bbox="727 248 861 274">? XXXnnnnn</pre> <p>donde <i>XXXnnnnn</i> representa un identificador válido de mensaje.</p> <p>Por ejemplo, ? SQL30081 muestra ayuda sobre el mensaje SQL30081.</p> <p>Para ver la ayuda sobre mensajes pantalla a pantalla, especifique:</p> <pre data-bbox="727 531 932 557">? XXXnnnnn   more</pre> <p>Para guardar la ayuda sobre el mensaje en un archivo, especifique:</p> <pre data-bbox="727 652 1085 678">? XXXnnnnn &gt; nombreamplio.ext</pre> <p>donde <i>nombreamplio.ext</i> representa el archivo en el que desea guardar la ayuda referente al mensaje.</p> |
| <i>Ayuda para SQL</i>      | Explica la sintaxis de las sentencias de SQL.                             | <p>Desde el procesador de línea de mandatos en modalidad interactiva, especifique:</p> <pre data-bbox="727 847 895 873">help sentencia</pre> <p>donde <i>sentencia</i> representa una sentencia de SQL.</p> <p>Por ejemplo, help SELECT visualiza ayuda sobre la sentencia SELECT.</p> <p><b>Nota:</b> En las plataformas basadas en UNIX no existe ayuda para SQL.</p>                                                                                                                                                                                                                                                                                                                              |
| <i>Ayuda para SQLSTATE</i> | Explica los estados y códigos de clase del SQL.                           | <p>Desde el procesador de línea de mandatos en modalidad interactiva, especifique:</p> <pre data-bbox="727 1150 1076 1177">? estado_sql o ? código_clase</pre> <p>donde <i>estado_sql</i> representa un estado SQL válido de cinco dígitos y <i>código_clase</i> representa los dos primeros dígitos del estado SQL.</p> <p>Por ejemplo, ? 08003 visualiza ayuda para el estado SQL 08003, mientras que ? 08 visualiza ayuda para el código de clase 08.</p>                                                                                                                                                                                                                                         |

## Visualización de información en línea

Los manuales que se incluyen con el presente producto están en copia software, en el formato HTML (Hypertext Markup Language). El formato en copia software le permite buscar o examinar información y proporciona enlaces de hipertexto con información afín. También facilita la utilización compartida de la biblioteca en el sitio Web.

Puede visualizar los manuales en línea o programas de ejemplo mediante cualquier navegador que cumpla las especificaciones de HTML Versión 3.2.

Para visualizar manuales en línea o programas de ejemplo:

- Si está ejecutando herramientas de administración de DB2, utilice el Centro de información.
- Desde un navegador, pulse **Archivo** —> **Abrir página**. La página que se abre contiene descripciones y enlaces que conducen a información sobre DB2.

- En las plataformas basadas en UNIX, abra la página siguiente:

```
INSTHOME/sql1lib/doc/%L/html/index.htm
```

donde %L representa el entorno nacional.

- En otras plataformas, abra la página siguiente:

```
sql1lib\doc\html\index.htm
```

La vía de acceso se encuentra en la unidad donde está instalado DB2.

Si no ha instalado el Centro de información, puede abrir la página efectuando una doble pulsación sobre el icono **Información de DB2**. Según cuál sea el sistema que esté utilizando, el icono se encuentra en la carpeta principal del producto o en el menú Inicio de Windows.

### Instalación del navegador Netscape

Si no tiene todavía un navegador Web instalado, puede instalar Netscape desde el CD-ROM proporcionado con el producto. Para obtener instrucciones detalladas sobre cómo instalarlo, siga los pasos siguientes:

1. Inserte el CD-ROM de Netscape.
2. Si utiliza una plataforma basada en UNIX, monte el CD-ROM. Consulte el manual *Guía rápida de iniciación* para conocer los procedimientos de montaje del CD-ROM.
3. Para obtener instrucciones sobre la instalación, consulte el archivo CDNAVnn.txt, donde nn representa el identificador de dos caracteres correspondiente a su idioma. El archivo está situado en el directorio raíz del CD-ROM.

## **Acceso a información mediante el Centro de información**

El Centro de información proporciona acceso rápido a información sobre los productos DB2. El Centro de información está disponible en todas las plataformas en las que pueden utilizarse las herramientas de administración de DB2.

Para abrir el Centro de información, haga una doble pulsación sobre su icono. Según cuál sea el sistema que esté utilizando, el icono se encuentra en la carpeta principal del producto o en el menú **Inicio** de Windows.

También puede acceder al Centro de información utilizando la barra de herramientas y el menú **Ayuda** en la plataforma Windows para DB2.

El Centro de información proporciona seis tipos de información. Pulse la pestaña adecuada para consultar el tipo de información correspondiente.

|                                |                                                                                                                                                                        |
|--------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Tareas</b>                  | Tareas esenciales que puede realizar mediante DB2.                                                                                                                     |
| <b>Consulta</b>                | Información de consulta sobre DB2, tal como palabras clave, mandatos y las API.                                                                                        |
| <b>Manuales</b>                | Manuales de DB2.                                                                                                                                                       |
| <b>Resolución de problemas</b> | Categorías de mensajes de error y sus acciones de recuperación.                                                                                                        |
| <b>Programas de ejemplo</b>    | Programas de ejemplo que se proporcionan con el DB2 Application Development Client. Si no instaló el DB2 Application Development Client, esta pestaña no se visualiza. |
| <b>Web</b>                     | Información sobre DB2 disponible en la World Wide Web. Para acceder a esta información, debe tener una conexión con la Web desde su sistema.                           |

Cuando selecciona un elemento de una de estas listas, el Centro de información abre un visor para mostrar la información. El visor puede ser el visor de ayuda del sistema, un editor o un navegador Web, dependiendo del tipo de información que seleccione.

El Centro de información proporciona una función de búsqueda, que le permite buscar un tema determinado sin examinar las listas.

Para realizar una búsqueda de texto completa, siga el enlace de hipertexto del Centro de información que conduce al formulario de búsqueda **Buscar información en línea sobre DB2**.

Normalmente, el servidor de búsqueda HTML arranca automáticamente. Si una búsqueda en la información HTML no funciona, puede que deba arrancar el servidor de búsqueda siguiendo uno de los métodos siguientes:

#### En Windows

Pulse **Inicio** y seleccione **Programas** —> **IBM DB2** —> **Información** —> **Iniciar servidor de búsqueda HTML**.

#### En OS/2

Haga una doble pulsación sobre la carpeta **DB2 para OS/2** y luego sobre el icono **Iniciar servidor de búsqueda HTML**.

Consulte las notas del release si tiene cualquier otro problema al buscar la información HTML.

**Nota:** La función de búsqueda no puede utilizarse en los entornos Linux, PTX ni Silicon Graphics IRIX.

### Utilización de los asistentes de DB2

Los asistentes ("wizards") le ayudan a realizar tareas de administración determinadas mediante instrucciones paso a paso. Puede acceder a los asistentes mediante el Centro de control y el Asistente de configuración de cliente. La tabla siguiente lista los asistentes y describe su función.

**Nota:** Los asistentes para Crear base de datos, Crear índice, Configurar actualización múltiple y Configuración del rendimiento pueden utilizarse en el entorno de base de datos particionada.

| Asistente                                        | Le ayuda a...                                                                                                  | Cómo acceder...                                                                                                                                                                             |
|--------------------------------------------------|----------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Añadir base de datos</i>                      | Catalogar una base de datos en una estación de trabajo cliente.                                                | En el Asistente de configuración del cliente, pulse <b>Añadir</b> .                                                                                                                         |
| <i>Hacer copia de seguridad de base de datos</i> | Determinar, crear y planificar un plan de copia de seguridad.                                                  | En el Centro de Control, pulse con el botón derecho del ratón sobre la base de datos que desea copiar y seleccione <b>Copia de seguridad</b> —> <b>Base de datos utilizando asistente</b> . |
| <i>Configurar actualización múltiple</i>         | Realizar una actualización múltiple, una transacción distribuida o una operación de confirmación de dos fases. | En el Centro de Control, pulse con el botón derecho del ratón sobre la carpeta <b>Bases de datos</b> y seleccione <b>Actualización múltiple</b> .                                           |
| <i>Crear base de datos</i>                       | Crear una base de datos y realizar algunas tareas básicas de configuración.                                    | En el Centro de Control, pulse con el botón derecho del ratón sobre la carpeta <b>Bases de datos</b> y seleccione <b>Crear</b> —> <b>Base de datos utilizando asistente</b> .               |

| <b>Asistente</b>                     | <b>Le ayuda a...</b>                                                                                                                                               | <b>Cómo acceder...</b>                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|--------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Crear tabla</i>                   | Seleccionar tipos de datos básicos y crear una clave primaria para la tabla.                                                                                       | En el Centro de Control, pulse con el botón derecho del ratón sobre el icono <b>Tablas</b> y seleccione <b>Crear</b> → <b>Tabla utilizando asistente</b> .                                                                                                                                                                                                                                                                                             |
| <i>Crear espacio de tablas</i>       | Crear un nuevo espacio de tablas.                                                                                                                                  | En el Centro de Control, pulse con el botón derecho del ratón sobre el icono <b>Espacios de tablas</b> y seleccione <b>Crear</b> → <b>Espacio de tablas utilizando asistente</b> .                                                                                                                                                                                                                                                                     |
| <i>Crear índice</i>                  | Determinar qué índices crear y eliminar para cada consulta.                                                                                                        | En el Centro de Control, pulse con el botón derecho del ratón sobre el icono <b>Índice</b> y seleccione <b>Crear</b> → <b>Índice utilizando asistente</b> .                                                                                                                                                                                                                                                                                            |
| <i>Configuración del rendimiento</i> | Ajustar el rendimiento de una base de datos actualizando los parámetros de configuración de acuerdo con sus necesidades.                                           | En el Centro de Control, pulse con el botón derecho del ratón sobre la base de datos que desea ajustar y seleccione <b>Configurar rendimiento utilizando asistente</b> .<br><br>Si utiliza un entorno de base de datos particionada, desde la vista Particiones de base de datos, pulse con el botón derecho del ratón sobre la primera partición de base de datos que desea ajustar y seleccione <b>Configurar rendimiento utilizando asistente</b> . |
| <i>Restaurar base de datos</i>       | Recuperar una base de datos después de una anomalía. Le ayuda a determinar qué copia de seguridad se debe utilizar y qué archivos de anotaciones se deben aplicar. | En el Centro de Control, pulse con el botón derecho del ratón sobre la base de datos que desea restaurar y seleccione <b>Restaurar</b> → <b>Base de datos utilizando asistente</b> .                                                                                                                                                                                                                                                                   |

## Configuración de un servidor de documentos

Por omisión, la información sobre DB2 se instala en el sistema local. Esto significa que cada una de las personas que deba acceder a la información sobre DB2 debe instalar los mismos archivos. Para que la información sobre DB2 se almacene en una única ubicación, siga los pasos siguientes:

1. Copie todos los archivos y subdirectorios del directorio `\sqllib\doc\html`, del sistema local, en un servidor Web. Cada manual tiene su propio subdirectorio que contiene todos los archivos HTML y archivos GIF necesarios que forman el manual. Asegúrese de que la estructura de directorios permanece igual.

2. Configure el servidor Web para que busque los archivos en la nueva ubicación. Si desea obtener más información, consulte el Apéndice sobre NetQuestion que se encuentra en la publicación *Suplemento de instalación y configuración*.
3. Si está utilizando la versión Java del Centro de información, puede especificar un URL base para todos los archivos HTML. Debe utilizar el URL para acceder a la lista de manuales.
4. Una vez que pueda visualizar los archivos del manual, puede marcar los temas que consulte con frecuencia. Probablemente deseará marcar las páginas siguientes:
  - Lista de manuales
  - Tablas de contenido de manuales utilizados con frecuencia
  - Temas consultados con frecuencia, tales como ALTERAR TABLA
  - El formulario de búsqueda

Para obtener información sobre cómo puede proporcionar los archivos de documentación en línea de DB2 Universal Database desde una máquina central, consulte el Apéndice sobre NetQuestion del manual *Suplemento de instalación y configuración*.

## Búsqueda de información en línea

Para buscar información en los archivos HTML, siga uno de los métodos siguientes:

- Pulse **Buscar** en el panel superior. Utilice el formulario de búsqueda para buscar un tema determinado. La función de búsqueda no puede utilizarse en los entornos Linux, PTX ni Silicon Graphics IRIX.
- Pulse **Índice** en el panel superior. Utilice el índice para buscar un tema determinado en el manual.
- Visualice la tabla de contenido o índice de la ayuda o del manual HTML y luego utilice la función de búsqueda del navegador Web para buscar un tema determinado en el manual.
- Utilice la función de marcaje de documentos del navegador Web para volver rápidamente a un tema determinado.
- Utilice la función de búsqueda del Centro de información para buscar temas determinados. Vea "Acceso a información mediante el Centro de información" en la página 1645 para obtener detalles.

---

## Apéndice S. Avisos

Es posible que IBM no comercialice en todos los países algunos productos, servicios o características descritos en este manual. Consulte al representante local de IBM para obtener información sobre los productos y servicios que actualmente pueden adquirirse en su zona geográfica. Cualquier referencia a un producto, programa o servicio de IBM no pretende afirmar ni implicar que sólo se puede utilizar dicho producto, programa o servicio de IBM. En su lugar se puede utilizar cualquier producto, programa o servicio funcionalmente equivalente que no infrinja ninguno de los derechos de propiedad intelectual de IBM. Sin embargo, es responsabilidad del usuario evaluar y verificar el funcionamiento de cualquier producto, programa o servicio que no sea de IBM.

IBM puede tener patentes o solicitudes de patentes en tramitación que afecten al tema tratado en este documento. La posesión de este documento no confiere ninguna licencia sobre dichas patentes. Puede realizar consultas sobre licencias escribiendo a:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
EE.UU.

Para realizar consultas sobre licencias referentes a información de doble byte (DBCS), puede ponerse en contacto con el Departamento de Propiedad Intelectual de IBM en su país o escribir a:

IBM World Trade Asia Corporation  
Licensing  
2-31 Roppongi 3-chome, Minato-ku  
Tokyo 106, Japón

**El párrafo siguiente no es aplicable al Reino Unido ni a ningún país en el que tales disposiciones sean incompatibles con la legislación local:**  
INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL", SIN GARANTÍA DE NINGUNA CLASE, NI EXPLÍCITA NI IMPLÍCITA, INCLUIDAS, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE NO VULNERACIÓN DE DERECHOS, COMERCIALIZABILIDAD O IDONEIDAD PARA UN FIN DETERMINADO. Algunos estados no permiten la exclusión de garantías expresas o implícitas en determinadas transacciones, por lo que es posible que esta declaración no sea aplicable en su caso.

Esta publicación puede contener inexactitudes técnicas o errores tipográficos. Periódicamente se efectúan cambios en la información aquí contenida; dichos cambios se incorporarán a las nuevas ediciones de la publicación. IBM puede efectuar, en cualquier momento y sin previo aviso, mejoras y/o cambios en los productos y/o programas descritos en esta publicación.

Las referencias hechas en esta publicación a sitios Web que no son de IBM se proporcionan sólo para la comodidad del usuario y no constituyen un aval de esos sitios Web. La información contenida en esos sitios Web no forma parte de la información del presente producto IBM y el usuario es responsable de la utilización de esos sitios Web.

IBM puede utilizar o distribuir cualquier información que se le facilite de la manera que considere adecuada, sin contraer por ello ninguna obligación con el remitente.

Los licenciatarios de este programa que deseen obtener información sobre él con el fin de habilitar: (i) el intercambio de información entre programas creados de forma independiente y otros programas (incluido este) y (ii) el uso mutuo de la información intercambiada, deben ponerse en contacto con:

IBM Canada Limited  
Office of the Lab Director  
1150 Eglinton Ave. East  
North York, Ontario  
M3C 1H7  
CANADÁ

Dicha información puede estar disponible, sujeta a los términos y condiciones apropiados, incluido en algunos casos, el pago de una tarifa.

El programa bajo licencia descrito en este manual y todo el material bajo licencia asociado a él, los proporciona IBM según los términos del Acuerdo de Cliente de IBM, el Acuerdo Internacional de Programas Bajo Licencia de IBM o cualquier acuerdo equivalente entre el usuario e IBM.

Los datos de rendimiento contenidos en este documento se obtuvieron en un entorno controlado. Por tanto, los resultados obtenidos en otros entornos operativos pueden variar significativamente. Algunas mediciones pueden haberse hecho en sistemas experimentales y no es seguro que estas mediciones sean las mismas en los sistemas disponibles comercialmente. Además, algunas mediciones pueden haberse calculado mediante extrapolación. Los resultados reales pueden variar. Los usuarios del presente manual deben verificar los datos aplicables para su entorno específico.

La información referente a productos que no son de IBM se ha obtenido de los proveedores de esos productos, de sus anuncios publicados o de otras



fuentes disponibles públicamente. IBM no ha probado esos productos y no puede confirmar la exactitud del rendimiento, la compatibilidad ni cualquier otra afirmación referente a productos que no son de IBM. Las preguntas sobre las prestaciones de productos que no son de IBM deben dirigirse a los proveedores de esos productos.

Todas las declaraciones de intenciones de IBM están sujetas a cambio o cancelación sin previo aviso, y sólo representan objetivos.

Esta publicación puede contener ejemplos de datos e informes que se utilizan en operaciones comerciales diarias. Para ilustrarlos de la forma más completa posible, los ejemplos incluyen nombres de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier similitud con nombres y direcciones utilizados por una empresa real es totalmente fortuita.

#### LICENCIA DE COPYRIGHT:

Este manual puede contener programas de aplicaciones de ejemplo escritos en lenguaje fuente, que muestran técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir estos programas de ejemplo de la forma que desee, sin pago alguno a IBM, con la intención de desarrollar, utilizar, comercializar o distribuir programas de aplicaciones de acuerdo con la interfaz de programación de aplicaciones correspondiente a la plataforma operativa para la que están escritos los programas de ejemplo. Estos ejemplos no se han probado exhaustivamente bajo todas las condiciones. Por tanto, IBM no puede asegurar ni implicar la fiabilidad, utilidad o función de estos programas.

Cada copia o porción de estos programas de ejemplo o cualquier trabajo derivado debe incluir una nota de copyright como la siguiente:

© (nombre de la empresa) (año). Partes de este código derivan de programas de ejemplo de IBM Corp. © Copyright IBM Corp. \_especifique el año o años\_. Reservados todos los derechos.

---

## Marcas registradas

Los términos siguientes, que pueden estar indicados por un asterisco (\*), son marcas registradas de International Business Machines Corporation en los Estados Unidos y/o en otros países.

|                                                 |                  |
|-------------------------------------------------|------------------|
| ACF/VTAM                                        | IBM              |
| AISPO                                           | IMS              |
| AIX                                             | IMS/ESA          |
| AIX/6000                                        | LAN DistanceMVS  |
| AIXwindows                                      | MVS/ESA          |
| AnyNet                                          | MVS/XA           |
| APPN                                            | Net.Data         |
| AS/400                                          | OS/2             |
| BookManager                                     | OS/390           |
| CICS                                            | OS/400           |
| C Set++                                         | PowerPC          |
| C/370                                           | QBIC             |
| DATABASE 2                                      | QMF              |
| DataHub                                         | RACF             |
| DataJoiner                                      | RISC System/6000 |
| DataPropagator                                  | RS/6000          |
| DataRefresher                                   | S/370            |
| DB2                                             | SP               |
| DB2 Connect                                     | SQL/DS           |
| DB2 Extenders                                   | SQL/400          |
| DB2 OLAP Server                                 | System/370       |
| DB2 Universal Database                          | System/390       |
| Distributed Relational<br>Database Architecture | SystemView       |
| DRDA                                            | VisualAge        |
| eNetwork                                        | VM/ESA           |
| Extended Services                               | VSE/ESA          |
| FFST                                            | VTAM             |
| First Failure Support Technology                | WebExplorer      |
|                                                 | WIN-OS/2         |

Los términos siguientes son marcas registradas de otras empresas:

Microsoft, Windows y Windows NT son marcas registradas de Microsoft Corporation.

Java, y las marcas registradas y logotipos basados en Java y Solaris, son marcas registradas de Sun Microsystems, Inc. en los Estados Unidos y/o en otros países.

Tivoli y NetView son marcas registradas de Tivoli Systems Inc. en los Estados Unidos y/o en otros países.

UNIX es una marca registrada en los Estados Unidos y/o en otros países bajo licencia exclusiva de X/Open Company Limited.

Otros nombres de empresas, productos o servicios, que pueden estar indicados por un doble asterisco (\*\*), pueden ser marcas registradas o marcas de servicio de otras empresas.



# Índice

## Caracteres Especiales

\* (asterisco)

en nombres de columna de selección 473

en nombres de columna de subselección 473

? (signo de interrogación)

marcador de parámetros

EXECUTE 1036

## A

ABS o ABSVAL, función  
descripción básica 236

descripción de formato  
detallado 280

valores y argumentos, reglas  
para 280

acceso a bases de datos

otorgar autorización 1056

acceso remoto

conexión IMPLICIT, diagrama de  
transición de estado 47

conexión no IMPLICIT, diagrama  
de transición de estado 49

conexión no satisfactoria 659

conexión satisfactoria 655

datos numéricos,  
conversiones 55

rol del servidor de

aplicaciones 44

sentencia CONNECT

EXCLUSIVE MODE, conexión  
dedicada 659

ON SINGLE NODE, conexión  
dedicada 659

SHARE MODE, de sólo  
lectura para ningún  
conector 659

sólo información de servidor,  
sin operandos 659

series de caracteres,  
conversiones 55

ACOS, función

descripción básica 236

descripción detallada 281

valores y argumentos 281

actualización posicional de columnas  
por fila 1196

ADD, cláusula

ALTER TABLE 571

ADD COLUMN, cláusula, orden de  
proceso 591

agrupaciones de almacenamientos  
intermedios

definición 74

descripción 39

establecimiento de tamaño 545,  
674

suprimir utilizando la sentencia  
DROP 1005

tamaño de página 675

utilización de almacenamiento  
ampliado 545, 675

agrupaciones de almacenamientos  
intermedios ampliados 546

ALIAS, cláusula

COMMENT, sentencia 630

DROP, sentencia 1008

almacenamiento

restitución, unidad de trabajo,  
ROLLBACK 1139

almacenamiento ampliado 675

ALTER, cláusula

sentencia GRANT (tabla, vista o  
apodo) 1071

sentencia REVOKE, eliminar  
privilegio 1132

ALTER BUFFERPOOL, sentencia  
descripción detallada 545

ALTER NODEGROUP, sentencia  
descripción detallada 552

ALTER SEQUENCE, sentencia  
descripción detallada 556

ALTER TABLE, sentencia  
autorización necesaria 565

diagrama de sintaxis 571

ejemplos 594

ALTER TABLESPACE, sentencia  
descripción detallada 597

ALTER TYPE (Estructurado),  
sentencia

descripción detallada 603

ALTER VIEW, sentencia  
autorización necesaria 613

descripción detallada 613

diagrama de sintaxis 613

ALL

predicado cuantificado 214

SELECT, sentencia 487

ALL, cláusula

SELECT, sentencia 473

ALL, opción

comparación, operador  
establecido 514

ALL PRIVILEGES, cláusula

REVOKE privilegios de tabla,  
vista o apodo 1132

sentencia GRANT (tabla, vista o  
apodo) 1071

ALLOCATE 1212

ALLOCATE CURSOR,  
sentencia 1212, 1213

ámbito

adición con sentencia ALTER  
TABLE 582

adición con sentencia ALTER  
VIEW 613

CREATE VIEW, sentencia 962  
definición 99

definición con columna  
añadida 574

definición con la sentencia  
CREATE TABLE 858

definición en especificación  
CAST 195

operación de desreferencia 196

AND, tabla de evaluación 232

antememoria

sentencia EXECUTE 1038

ANY

predicado cuantificado 214

añadir base de datos, asistente  
para 1646, 1647

apodo 58

nombre, sintaxis 75

apodos

calificación de un nombre de  
columna 144

cláusula FROM 478

cláusula FROM, convenios de  
denominación de  
subselección 478

cláusula SELECT, diagrama de  
sintaxis 473

descripción detallada 796

- apodos (*continuación*)
  - nombres expuestos en cláusula FROM 145
  - nombres no expuestos en cláusula FROM 145
  - otorgar privilegio de control 1072
  - otorgar privilegios 1070
  - revocar privilegios 1131
- archivo, variables de referencia
  - BLOB 155
  - CLOB 155
  - DBCLOB 155
- área SQLDA, variables necesarias para DESCRIBE 997
- argumentos de COALESCE
  - tipo de datos de resultado 120
- aritmética
  - AVG, operación de la función 258
  - buscar valor máximo 268
  - buscar valor mínimo 270
  - columnas, adición de valores (SUM) 277
  - coma flotante, rango y precisión 90
  - constantes
    - definición de 129
    - NOT NULL, atributo necesario 129
  - devolución de valores de enteros pequeños de expresiones 421
  - entero
    - entero grande, rango y precisión 90
    - entero pequeño, rango y precisión 90
  - expresiones, adición de valores (SUM) 277
  - fecha y hora, reglas de SQL 185
  - funciones de regresión 272
  - marcadores de parámetros 1101
  - operación de función VARIANCE 278
  - operación de la función CORRELATION 260
  - operación de la función COVARIANCE 265
  - operaciones de fecha, reglas 188
  - operaciones de fecha y hora, reglas 189
  - operaciones de hora, reglas 188, 189
  - operaciones decimales, fórmulas de escala y precisión 182
- aritmética (*continuación*)
  - operandos de coma flotante con enteros, resultados 183
  - reglas y valores de precisión 183
  - operandos de tipo diferenciado 183
  - signo más unitario, efecto en el operando 181
  - signo menos unitario, efecto en el operando 181
  - STDDEV, función 276
  - uso remoto de, conversiones 55
  - valor decimal, precisión y escala 90
  - valores de coma flotante de expresiones numéricas 329, 404
  - valores decimales de expresiones numéricas 311
  - valores enteros, devolución de expresiones 286, 353
- aritméticos
  - operadores, resumen 180
- Arquitectura de bases de datos relacionales distribuidas (DRDA)
  - definición 43
- AS, cláusula
  - CREATE VIEW, sentencia 960
  - en cláusula SELECT 473, 476
  - ORDER BY, cláusula 522
- ASC, cláusula
  - CREATE INDEX, sentencia 777
  - sentencia select 524
- ASCII, función
  - descripción básica 237
  - descripción detallada 282
  - valores y argumentos 282
- asignación de una serie a una columna, reglas para 107
- asignaciones
  - almacenamiento 108, 194
  - fragmentación de un carácter
    - MBCS, reglas para 109
  - números 107
  - recuperación 108
  - reglas de serie 107
  - relleno con blancos de series de caracteres mixtos 109
  - serie de caracteres mixtos para variables del lenguaje
    - principal 109
  - series de caracteres a columnas de indicación de fecha y hora, reglas para 105
- asignaciones (*continuación*)
  - tipo DATALINK 111
  - tipo de referencia 114
  - tipo definido por el usuario 113
  - truncamiento de series de caracteres mixtos 109
  - valor de indicación de fecha y hora a serie de caracteres 110
  - valores de indicación de fecha y hora, reglas para 110
- ASIN, función
  - descripción básica 237
  - descripción detallada 283
  - valores y argumentos 283
- asistente
  - restaurar base de datos 1647
- asistentes
  - añadir base de datos 1646, 1647
  - configuración del rendimiento 1647
  - configurar actualización múltiple 1646
  - copiar base de datos 1646
  - crear base de datos 1646
  - crear espacio de tablas 1647
  - crear tabla 1646
  - índice 1647
  - realización de tareas 1646
- ASSOCIATE LOCATORS,
  - sentencia 1216, 1217
- asterisco (\*)
  - en COUNT 261
  - en COUNT\_BIG 263
  - en nombres de columna de selección 473
  - en nombres de columna de subselección 473
- ATAN, función
  - descripción básica 237
  - descripción detallada 284
  - valores y argumentos 284
- ATAN2, función
  - descripción básica 237
  - descripción detallada 285
  - valores y argumentos 285
- atributos de longitud de columnas 87
- autorización
  - control público en índices 1059
  - creación pública en secuencia 1067
  - crear público en esquema 1064
  - definición 10
  - otorgar control sobre índices 1059

- autorización (*continuación*)
    - otorgar creación en secuencia 1067
    - otorgar el control en operaciones de bases de datos 1056
    - otorgar para crear en esquema 1064
  - autorización
    - IMPLICIT\_SCHEMA 13
  - autorizaciones
    - revocar 1118
  - autorreferentes, filas 20
  - autorreferentes, tablas 20
  - AVG, descripción detallada de la función 258
  - AVG, función
    - descripción básica 237
  - ayuda en línea 1642
  - B**
  - base de datos de muestra
    - borrado 1428
    - creación 1428
    - tablas 1427
  - base de datos relacional distribuida
    - entorno 44
    - petionario de aplicaciones 43
    - protocolos de
      - petionario-servidor 43
    - representación de datos
      - consideraciones 55
    - servidor de aplicaciones 43
    - unidad de trabajo remota 45
  - base de datos relacional distribuida, definición 43
  - bases de datos relacionales
    - definición 9
  - BEGIN DECLARE SECTION, sentencia
    - autorización necesaria 615
    - descripción detallada 615
    - reglas de invocación 615
  - biblioteca de DB2
    - asistentes 1646
    - ayuda en línea 1642
    - buscar información en línea 1648
    - Centro de información 1645
    - configuración de un servidor de documentos 1647
    - estructura de 1629
    - identificador de idioma para manuales 1638
    - imprimir manuales PDF 1640
  - biblioteca de DB2 (*continuación*)
    - información de última hora 1639
    - manuales 1629
    - pedido de manuales impresos 1640
    - visualización de información en línea 1644
  - BIGINT
    - descripción 90
    - precisión 90
    - rango 90
    - tipo de datos 90
  - BIGINT, función
    - descripción básica 237
    - valores enteros de expresiones 286
  - BIGINT, tipo de datos 850
  - BINDADD, parámetro
    - otorgar privilegio 1056
  - BLOB
    - tipo de datos 851
  - BLOB, función
    - descripción básica 237
  - bloqueo
    - COMMIT, efecto de la sentencia sobre 641
    - definición 27
    - filas y columnas de tabla, restricción del acceso 1092
    - LOCK TABLE, sentencia 1092
  - bloqueos
    - durante UPDATE 1199
  - INSERT, reglas por omisión para la sentencia 1090
  - modalidad de actualización (U) 24
  - modalidad de compartimiento (S) 24
  - modalidad de exclusividad (X) 24
  - tablas temporales declaradas, falta de 24
  - terminación para unidad de trabajo, ROLLBACK 1139
- bloques de actualización 24
  - bloques de compartimiento 24
  - bloques de exclusividad 24
  - borrado
    - base de datos de muestra 1428
  - BUFFERPOOL, cláusula
    - ALTER TABLESPACE, sentencia 599
    - CREATE TABLESPACE, sentencia 899
  - BUFFERPOOL, cláusula (*continuación*)
    - DROP, sentencia 1008
  - buscar
    - información en línea 1645, 1648
  - C**
  - calificación de nombre de columna en sentencia COMMENT ON 144
  - calificador
    - reservado 1447
  - calificados, nombres de columna
    - reglas 144
  - CALL, sentencia
    - descripción detallada 617
  - cambios no confirmados, relación con los bloqueos 27
  - campo SQLD en SQLDA
    - descripción 1269
  - campo SQLDABC en SQLDA
    - descripción 1269
  - campo SQLDAID en SQLDA
    - descripción 1269
  - campo SQLDATALEN en SQLDA
    - descripción 1273
  - campo SQLDATATYPE\_NAME en SQLDA
    - descripción 1274
  - campo SQLIND en SQLDA
    - descripción 1271
  - campo SQLN en SQLDA
    - descripción 1269
  - campo SQLNAME en SQLDA
    - descripción 1272
  - campo SQLTYPE en SQLDA
    - descripción 1270
  - campo SQLVAR en SQLDA
    - base 1270
    - secundaria 1272
  - campo SQLLEN en SQLDA
    - descripción 1270
  - campo SQLLONGLEN en SQLDA
    - descripción 1272
  - cancelación de una unidad de trabajo 1139
  - carácter de desplazamiento a teclado estándar
    - no truncado por asignaciones 110
  - carácter en blanco
    - definición 69
  - caracteres
    - SQL, rango 69
  - caracteres comodín
    - valores en predicado LIKE 223

- caracteres de doble byte truncados durante la asignación 110
- caracteres de escape, SQL 72
- caracteres especiales
  - blanco 70
  - espacio 70
  - rango 70
- cargar una base de datos, otorgar autorización para 1057
- CASCADE, regla de supresión 875
  - descripción 22
- CASE, sentencia 1218
- CAST
  - especificaciones 193
  - expresión como operando 193
  - marcador de parámetros como operando 194
  - null como operando 194
- catálogos
  - adición de comentarios sobre tablas, vistas, columnas 628
- COMMENT, sintaxis detallada 628
- CEIL, función
  - descripción detallada 288
  - valores y argumentos 288
- CEIL o CEILING, función
  - descripción básica 237
- CEILING, función
  - descripción detallada 288
  - valores y argumentos 288
- Centro de información 1645
- cifrar información
  - función ENCRYPT 331
  - función GETHINT 338
- CL\_SCHED, tabla de muestra 1429
- clasificación
  - clasificación de los resultados 117
  - comparaciones de series 115
- cláusula BETWEEN
  - utilizar en funciones OLAP 197
- cláusula-contenedor
  - CREATE TABLESPACE, sentencia 897
- cláusula CONTINUE
  - WHENEVER, sentencia 1207
- cláusula CHECK en la sentencia
  - CREATE VIEW 964
- cláusula-except-on-nodes
  - CREATE BUFFERPOOL, sentencia 675
- cláusula FROM, uso y ejemplo de nombre-correlación 144
- cláusula-on-nodes
  - CREATE TABLESPACE, sentencia 896, 898
- cláusula OVER
  - utilizar en funciones OLAP 197
- cláusula PARTITION BY
  - utilizar en funciones OLAP 197
- cláusula RANGE
  - utilizar en funciones OLAP 197
- cláusula ROW
  - utilizar en funciones OLAP 197
- cláusula VALUES de múltiples filas
  - tipo de datos de resultado 120
- cláusulas ESCAPE
  - LIKE, predicado 225
- clave de inicio 786
- clave de parada 786
- clave de particionamiento
  - ALTER TABLE, sentencia 581
  - consideraciones 882
  - definición al crear tabla 871
- clave de unicidad
  - ALTER TABLE, sentencia 575
- clave foránea
  - nombre de restricción, convenios para 875
  - vista, restricciones de referencia 15
- claves
  - compuestas 17
  - de unicidad 17, 18
  - exclusivas 17
  - foránea 19
  - foráneas 17
  - padre 20
  - partición 17
  - primarias 17
- claves de particionamiento
  - adición con ALTER TABLE 565
  - eliminación con ALTER TABLE 565
- claves foráneas
  - adición con ALTER TABLE 565
  - eliminación con ALTER TABLE 565
- claves primarias
  - adición con ALTER TABLE 565
  - eliminación con ALTER TABLE 565
  - otorgar privilegios de adición 1071
  - otorgar privilegios de eliminación 1071
- CLI
  - definición 31
- CLIENT ACCTNG, registro especial 132
- CLIENT APPLNAME, registro especial 132
- CLIENT USERID, registro especial 133
- CLIENT WRKSTNNAME, registro especial 133
- cliente/servidor
  - nombre de servidor, descripción 77
- CLOB, función
  - descripción básica 238
  - descripción detallada 296
  - valores y argumentos 296
- CLOB, tipo de datos 852
- CLOSE, sentencia 626, 627
- CLUSTER, cláusula
  - CREATE INDEX, sentencia 778
- COALESCE, función
  - descripción básica 238
  - descripción detallada 297
- códigos de error
  - definiciones de SQLCA 1261
- códigos de retorno
  - sentencias de SQL ejecutables 539
  - sentencias incorporadas 542
- códigos de retorno de aviso 542
- códigos de retorno de SQL
  - visión general 542
- COLUMN, cláusula
  - COMMENT, sentencia 630
- columna
  - nombre de restricción, FOREIGN KEY, reglas 875
- columna de OID 868
- columnas
  - actualización de los valores de columna, sentencia UPDATE 1193
  - adición con sentencia ALTER TABLE 565
  - adición de comentarios al catálogo 628
  - adición de valores (SUM) 277
  - añadir a una tabla, ALTER TABLE 571
  - básico, uso en series coincidentes del predicado 213
  - buscar valor máximo 268
  - buscar valor mínimo 270
  - búsqueda utilizando la cláusula WHERE 486



- columnas (*continuación*)
  - cláusula HAVING, nombres de búsqueda, reglas 495
  - cláusula SELECT, diagrama de sintaxis 473
  - convenios de denominación, aplicaciones
    - en expresiones 143
    - en sentencia CREATE INDEX 143
    - en sentencia CREATE TABLE 143
    - en sentencias GROUP BY 143
    - en sentencias ORDER BY 143
  - correlación entre un conjunto de pares de números (CORRELATION) 260
  - covarianza de un conjunto de pares de números (COVARIANCE) 265
  - crear claves de índice 776
  - datos de resultado 477
  - definición 14
  - desviación estándar de un conjunto de valores (STDDEV) 276
  - errores de referencia de nombre ambigua 148
  - errores de referencia de nombre no definida 148
  - escalar, selección completa 149
  - expresión de tabla anidada 149
  - GROUP BY, utilización para limitar la cláusula SELECT 475
  - HAVING, utilización para limitar la cláusula SELECT 475
  - inserción de valores, sentencia INSERT 1083
  - nombre, condiciones calificadas 150
  - nombre, condiciones no calificadas 150
  - nombres de columnas de agrupación en GROUP BY 487
  - nombres en cláusula ORDER BY 523
  - otorgar privilegios de adición 1071
  - palabra clave DISTINCT, consultas 257
  - predicado BETWEEN, en series coincidentes 217
- columnas (*continuación*)
  - predicado EXISTS, en series coincidentes 219
  - predicado IN, selección completa, valores devueltos 220
  - predicado LIKE, en series coincidentes 223
  - promedio de un conjunto de valores (AVG) 258
  - reglas de asignación de serie 107
  - reglas para nombre de columna calificado 144
  - series de caracteres de longitud fija, atributos 87
  - series de caracteres de longitud variable, atributos 87
  - subconsulta 149
  - valores nulos, ALTER TABLE, prevención 574
  - valores nulos en columnas de resultado 475
  - varianza de un conjunto de valores de columna (VARIANCE) 278
- columnas del resultado de subselección 476
- columnas generadas
  - CREATE TABLE, sentencia 861
- coma flotante de doble precisión 90
- coma flotante de precisión simple 90
- coma flotante de precisión simple, tipo de datos 850
- combinación de conjuntos de agrupación 492
- comentarios
  - en tabla del catálogo 628
  - lenguaje principal, formato 71
  - sentencias de SQL estáticas 544
  - SQL, formato 71
- comm\_rate, opción de servidor 1417
- COMMENT, sentencia descripción detallada 628
- COMMIT, sentencia descripción detallada 640
- paso a través 1424
- comparación
  - reglas de compatibilidad 105
  - reglas de compatibilidad, tipos de datos, resumen 105
  - tipo definido por el usuario 119
- comparación de dos predicados, condiciones verdaderas 213, 230
- comparación de series LONG VARCHAR, uso restringido 118
- comparación de un valor con una selección 217
- comparaciones
  - LONG VARCHAR, uso restringido 118
  - números 114
  - SBCS/MBCS, reglas 117
  - series, reglas 115
  - series gráficas, reglas 118
  - tipo de referencia 120
  - valores de indicación de fecha y hora, reglas 119
- compatibilidad
  - reglas 105
  - reglas para los tipos de operaciones 105
  - tipos de datos 105
  - tipos de datos, resumen 105
- compatibilidad entre particiones definición 127
- compensación 58
- compuestas, claves definición 17
- CONCAT, función
  - descripción detallada 298
  - valores y argumentos 298
- CONCAT o ||, función descripción básica 238
- concatenación
  - longitud del resultado 179
  - operadores 177
  - tipo de datos del resultado 179
  - tipo diferenciado 180
- condición desconocida valor nulo 232
- condiciones-búsqueda
  - con DELETE, selección de filas 993
- condiciones de búsqueda
  - AND, operador lógico 232
  - cláusula HAVING, argumentos y reglas 495
  - cláusula WHERE 486
  - con UPDATE, aplicar cambios 1198
  - descripción 232
  - NOT, operador lógico 232
  - OR, operador lógico 232
  - orden de evaluación 233
- configuración de un servidor de documentos 1647

- configuración del rendimiento, asistente de 1647
- configurar actualización múltiple, asistente para 1646
- conjuntos-agrupación 488
- conjuntos de agrupación
  - ejemplos 503
- conjuntos resultantes
  - devolución desde un procedimiento SQL 1236
- CONNECT, parámetro en sentencia GRANT...ON DATABASE 1056
- CONNECT, sentencia (Tipo 1)
  - descripción detallada 653
  - EXCLUSIVE MODE 653
  - SHARE MODE 653
- CONNECT, sentencia (Tipo 2)
  - descripción detallada 662
- CONNECT TO, sentencia
  - conexión no satisfactoria 659
  - conexión no satisfactoria, descripción detallada 663
  - conexión satisfactoria 655
  - conexión satisfactoria, descripción detallada 662
- constantes
  - con tipos definidos por el usuario 131
  - decimal 130
  - definición de entero 129
  - hexadecimal 130
  - reglas para coma flotante 129
  - serie de caracteres, rango y precisión 130
  - visión general 129
- constantes de coma flotante 129
- CONSTRAINT, cláusula
  - COMMENT, sentencia 630
- consulta
  - ejemplo 533
- consulta de SQL
  - subconsulta, cláusula WHERE 486
- consultas 471
  - definición 26, 471
  - ejemplo de recurrencia 1499
  - ID de autorización necesarios 471
  - recursiva 520
- contenedores
  - CREATE TABLESPACE, sentencia 896
  - descripción 39
- contenedores-basedatos
  - CREATE TABLESPACE, sentencia 897
- contenedores-sistema
  - CREATE TABLESPACE, sentencia 896
- CONTROL, cláusula
  - sentencia GRANT (tabla, vista o apodo) 1072
- CONTROL, cláusula en sentencia GRANT, revocación 1132
- CONTROL, parámetro
  - revocar privilegios para paquetes 1125
- convenios de denominación
  - columnas 74
  - etiquetas 75
  - reglas para columna calificada 144
  - SQL 72
- conversión
  - entre tipos de datos 101
  - tipos de referencia 102
  - tipos definidos por el usuario 102
- conversión de caracteres
  - elemento de código 34
  - esquemas de codificación 34
  - juegos de caracteres 34
  - páginas de códigos 34
  - reglas al comparar series 124
  - reglas para asignaciones 108
  - reglas para la comparación 117
  - reglas para operaciones que combinan series 124
- conversión de coma flotante a decimal 107
- conversión decimal de entero, resumen 107
- conversiones
  - CHAR, devolución de valores de indicación de fecha y hora convertidos 289
  - DBCS de SBCS y DBCS mezclados 452
  - de serie de caracteres a indicación de fecha y hora 436
  - entero a decimal, reglas para expresión mixta 182
  - fecha y hora a variable de serie de caracteres 110
  - numérico, escala y precisión, resumen 107
  - serie de caracteres a SQL ejecutable 1042
- conversiones (*continuación*)
  - serie de caracteres de doble byte 452
  - valores de coma flotante de expresiones numéricas 329, 404
  - valores decimales de expresiones numéricas 311
- copiar base de datos, asistente para 1646
- correlación de funciones 58
- correlación de tipos de datos 57
- correlación de usuarios 57
- correlaciones de funciones
  - descripción del nombre 74
- correlaciones de particiones
  - creación para grupos de nodos 802
- correlaciones de tipos
  - convenios de nombre 77
  - descripción del nombre 77
- CORRELATION, función
  - descripción detallada 260
- CORRELATION o CORR 238
- COS, función
  - descripción básica 238
  - descripción detallada 299
  - valores y argumentos 299
- COT, función
  - descripción básica 238
  - descripción detallada 300
  - valores y argumentos 300
- COUNT, función
  - descripción básica 238
  - descripción de formato detallado 261
  - valores y argumentos 261
- COUNT\_BIG, función
  - descripción básica 239
  - descripción de formato detallado 263
  - valores y argumentos 263
- COVARIANCE, función
  - descripción detallada 265
- creación
  - base de datos de muestra 1428
- creación de un valor DATALINK DLVALUE, función 327
- creación de una base de datos, otorgar autorización para 1057
- crear base de datos, asistente para 1646
- crear espacio de tablas, asistente para 1647
- crear tabla, asistente para 1646

CREATE ALIAS, sentencia  
descripción detallada 670

CREATE BUFFERPOOL, sentencia  
cláusula-except-on-nodes 675  
descripción detallada 673

CREATE DISTINCT TYPE, sentencia  
descripción detallada 677

CREATE EVENT MONITOR,  
sentencia  
descripción detallada 684

CREATE FUNCTION (Escalar  
externa), sentencia 696

CREATE FUNCTION (fuente o  
plantilla), sentencia 750

CREATE FUNCTION (SQL, escalar,  
de tabla o de fila), sentencia 761

CREATE FUNCTION (Tabla  
externa), sentencia 723

CREATE FUNCTION MAPPING,  
sentencia  
descripción detallada 769

CREATE INDEX, sentencia  
descripción detallada 774  
nombres-columna en claves de  
índice 776

CREATE INDEX EXTENSION,  
sentencia 782

CREATE METHOD, sentencia 790

CREATE NICKNAME, sentencia  
descripción detallada 795

CREATE NODEGROUP, sentencia  
descripción detallada 801

CREATE PROCEDURE, sentencia  
DECLARE, sentencia 1233  
descripción detallada 804  
gestores de condiciones 1236  
sentencia CASE 1218  
sentencia compuesta de  
procedimiento 1233  
sentencia de asignación 1214  
sentencia de gestor de  
condiciones 1236  
sentencia de procedimiento  
SQL 1210  
sentencia dinámica  
compuesta 642  
sentencia FOR 1221  
sentencia GET  
DIAGNOSTICS 1223  
sentencia GOTO 1225  
sentencia IF 1227  
sentencia ITERATE 1229  
sentencia LEAVE 1230  
sentencia LOOP 1231  
sentencia REPEAT 1240

CREATE PROCEDURE, sentencia  
(*continuación*)  
sentencia RESIGNAL 1242  
sentencia RETURN 1245  
sentencia SIGNAL 1248  
sentencia WHILE 1251  
variables 1233

CREATE SCHEMA, sentencia  
descripción detallada 822

CREATE SEQUENCE, sentencia  
descripción detallada 826

CREATE TABLE, sentencia 835, 892  
diagrama de sintaxis 836

CREATE TABLESPACE, sentencia  
descripción detallada 893

CREATE TRANSFORM, sentencia  
descripción detallada 903

CREATE TRIGGER, sentencia  
descripción detallada 910

CREATE TYPE (Estructurado),  
sentencia 923, 949

CREATE VIEW, sentencia  
descripción detallada 957

CREATETAB, parámetro de  
sentencia GRANT...ON  
DATABASE 1056

CUBE  
descripción de consulta 491  
ejemplos 503

CURRENT DATE, registro  
especial 134

CURRENT DEFAULT TRANSFORM  
GROUP, registro especial 134

CURRENT DEGREE, registro  
especial 135  
SET CURRENT DEGREE,  
sentencia 1151

CURRENT EXPLAIN MODE,  
registro especial 136  
SET CURRENT EXPLAIN  
MODE, sentencia 1153

CURRENT EXPLAIN SNAPSHOT,  
registro especial 137  
SET CURRENT EXPLAIN  
SNAPSHOT, sentencia 1155

CURRENT FUNCTION PATH,  
registro especial 138  
SET CURRENT FUNCTION  
PATH, sentencia 1181  
SET CURRENT PATH,  
sentencia 1181  
SET PATH, sentencia 1181

CURRENT NODE, registro  
especial 138

CURRENT PATH, registro  
especial 138  
SET CURRENT FUNCTION  
PATH, sentencia 1181  
SET CURRENT PATH,  
sentencia 1181  
SET PATH, sentencia 1181

CURRENT QUERY OPTIMIZATION,  
registro especial 139  
SET CURRENT QUERY  
OPTIMIZATION,  
sentencia 1159

CURRENT REFRESH AGE, registro  
especial 140  
sentencia SET CURRENT  
REFRESH AGE 1162

CURRENT SCHEMA, registro  
especial 140

CURRENT SERVER, registro  
especial 141

CURRENT SQLID, registro  
especial 140

CURRENT TIME, registro  
especial 141

CURRENT TIMESTAMP, registro  
especial 142

CURRENT TIMEZONE, registro  
especial 142

cursor  
cierre, sentencia CLOSE 626

CURSOR FOR RESULT SET 1212

cursores  
ambiguo 980  
apertura de un cursor, sentencia  
OPEN 1094  
conjunto activo, asociación 1094  
declarar, sintaxis de sentencias de  
SQL 976  
definición 976  
determinación de la posibilidad  
de actualización 979  
estado cerrado, condiciones  
previas 1097  
estado de sólo lectura,  
condiciones 979  
fila actual 1052  
movimiento de la posición,  
utilización de FETCH 1050  
posiciones de abierto 1052  
preparación para que lo utilice la  
aplicación 1094  
relación con tabla resultante 976  
supresión, detalles de condición  
de búsqueda 994

- cursores (*continuación*)
    - terminación para unidad de trabajo, ROLLBACK 1139
    - ubicación en la tabla, resultados de FETCH 1050
    - unidad de trabajo, estados condicionales 976
    - utilización del programa 979
    - WITH HOLD, cláusula de bloqueo de la sentencia COMMIT, efecto 641
  - cursores ambiguos 980
  - cursores de sólo lectura
    - ambiguo 980
- CH**
- CHAR
  - función descripción 289
- CHAR, función
  - descripción básica 238
- CHAR VARYING, tipo de datos 851
- CHARACTER VARYING, tipo de datos 851
- CHR, función
  - descripción básica 238
  - descripción detallada 295
  - valores y argumentos 295
- D**
- DATE
  - función escalar WEEK 454
  - función escalar WEEK\_ISO 455
- DATE, función
  - conversión de formato de valor a fecha 301
  - descripción básica 239
  - descripción detallada 301
  - operaciones aritméticas 186
- DATE, tipo de datos 852
- datos de bit
  - definición 88
  - serie BLOB 85
- datos MBCS (juego de caracteres de doble byte)
  - dentro de datos mixtos 88
- datos mixtos
  - descripción 88
  - LIKE, predicado 225
- datos numéricos
  - conversiones remotas 55
- datos SBCS (juego de caracteres de un solo byte)
  - dentro de datos mixtos 88
  - descripción 88
- DAY, función
  - descripción básica 239
  - devolución de la parte correspondiente al día de los valores 303
- DAYNAME, función
  - descripción básica 239
  - descripción detallada 305
  - valores y argumentos 305
- DAYOFWEEK, función
  - descripción básica 239
  - descripción detallada 306
  - valores y argumentos 306
- DAYOFWEEK\_ISO, función
  - descripción básica 239
  - descripción de formato detallado 307
  - valores y argumentos, reglas para 307
- DAYOFYEAR, función
  - descripción básica 239
  - descripción detallada 308
  - valores y argumentos 308
- DAYS, función
  - descripción básica 240
  - devolución de las duraciones de entero 309
- db2nodes.cfg
  - ALTER NODEGROUP 553
  - CONNECT (Tipo 1) 660
  - CREATE NODEGROUP 801
  - CURRENT NODE 138
  - NODENUMBER, función 390
- DBADM, parámetro, sentencia GRANT...ON DATABASE 1057
- DBCLOB, función
  - descripción básica 240
  - descripción detallada 310
  - valores y argumentos 310
- DBCLOB, tipos de datos 852
- DDL
  - definición 9, 58
- decimal
  - coma decimal implícita 90
  - constantes, rango y precisión 130
  - conversión de coma flotante 107
  - decimal empaquetado 90
  - fórmulas aritméticas, escala y precisión 182
  - números 90
  - tipo de datos, visión general 90
  - tipos de datos 83
- DECIMAL, función
  - descripción detallada 311
- DECIMAL, función (*continuación*)
  - valores y argumentos 311
- DECIMAL o DEC, función
  - descripción básica 240
- declaraciones
  - inserción en un programa 1080
- DECLARE
  - BEGIN DECLARE SECTION, sentencia 615
  - END DECLARE SECTION, sentencia 1034
- DECLARE, sentencia 1233
- DECLARE CURSOR, sentencia 976
  - condiciones para la autorización 976
  - descripción detallada 976
  - notas de utilización del programa 979
- DECLARE GLOBAL TEMPORARY TABLE, sentencia
  - descripción detallada 982
- DECRYPT, función
  - descripción detallada 314
  - valores y argumentos 314
- DECRYPT\_BIN, función
  - descripción básica 240
- DECRYPT\_CHAR, función
  - descripción básica 240
- definición de servidor 57
- definición de tabla
  - ADVISE\_WORKLOAD creación 1493
- definida por el usuario, función CREATE FUNCTION (Escalar externa), sentencia 696
- CREATE FUNCTION (Tabla externa), sentencia 723
- sentencia CREATE FUNCTION (fuente o plantilla) 750
- DEGREES, función
  - descripción básica 240
  - descripción detallada 316
  - valores y argumentos 316
- DELETE, cláusula
  - sentencia GRANT (tabla, vista o apodo) 1072
  - sentencia REVOKE, revocar privilegios 1132
- DELETE, sentencia
  - autorización, formato buscado o posicionado 991
  - descripción detallada 991
- delimitado, identificador de SQL 71
- delimitadores, definición de símbolos 70

DENSE\_RANK  
función OLAP 197

DENSERANK  
función OLAP 197

DEPARTMENT, tabla de muestra 1429

dependencia  
de objetos entre sí 1023

dependientes, filas 20

dependientes, tablas 20

desagrupación parcial 42

DESC, cláusula  
CREATE INDEX, sentencia 777  
de la sentencia select 524

descendientes, tablas 20

descendientes filas 20

descifrar información  
función DECRYPT 314

DESCRIBE, sentencia  
descripción detallada 997  
sentencias preparadas, condiciones de destrucción 999

DESCRIPTOR  
variables del lenguaje principal, lista de sustitución de parámetros 1037

desencadenantes  
acciones desencadenadas 38  
activación 37  
adición de comentarios al catálogo 628  
cascada 38  
convenios de nombre 77  
CREATE TRIGGER, sentencia 910  
descripción 37  
descripción del nombre 77  
DROP, sentencia 1020  
filas afectadas 38  
granularidad 38  
INSERT, sentencia 1086  
interacciones 1455  
mensajes de error 918  
momento de activación 38  
no operativo 917  
restricciones 1455  
sucesos 38  
tablas con tipo 918  
Tablas de Explain 1459  
tablas sujeto 37  
usos 37

desencadenantes no operativos  
descripción detallada 917

devolución de conjuntos resultantes 1236

devolución de la indicación de la fecha y hora de valores  
TIMESTAMP, función 436

devolución de la parte correspondiente a la hora de los valores  
HOUR, función 344

devolución de mes del valor  
MONTH, función 373

devolución de microsegundo del valor  
MICROSECOND, función 369

devolución de minuto del valor  
MINUTE, función 371

devolución de segundos del valor  
SECOND, función 418

devolución de subseries de una serie  
SUBSTR, función 425

devolver valor de columna de identidad  
función  
IDENTITY\_VAL\_LOCAL 345

día del valor  
DAY, función 303

diagramas de sintaxis  
descripción 3

DIFFERENCE, función  
descripción básica 240  
descripción detallada 318  
valores y argumentos 318

dígitos  
rango 70

DIGITS, función  
descripción básica 241  
descripción detallada 319  
valores y argumentos 319

dinámica  
selección de marcadores de parámetros 540  
selección de variable del lenguaje principal 540

dinámica, sentencia compuesta 642

dinámicas, sentencias de SQL  
métodos de preparación 538

DISCONNECT, sentencia  
descripción detallada 1002

disminuir una fecha, reglas 187

disminuir una hora, reglas 189

DISTINCT, cláusula  
de subselección 473

DISTINCT, palabra clave  
AVG, relación con la función 258

DISTINCT, palabra clave  
(continuación)  
función de columna 257  
función STDDEV, relación 276

MIN, función 270  
relación con función  
COUNT 261  
relación con función  
COUNT\_BIG 263  
relación con función  
VARIANCE 278  
restricción de función MAX 268  
SUM, función 277

DISTINCT, palabra clave, visión general 257

DISTINCT TYPE, cláusula  
DROP, sentencia 1020  
sentencia COMMENT 637

DLCOMMENT, función  
descripción básica 241  
descripción detallada 320  
valores y argumentos 320

DLURLCOMPLETE, función  
descripción básica 241  
descripción detallada 322  
valores y argumentos 322

DLURLPATH, función  
descripción básica 241  
descripción detallada 323  
valores y argumentos 323

DLURLPATHONLY, función  
descripción básica 241  
descripción detallada 324  
valores y argumentos 324

DLURLSCHEME, función  
descripción básica 241  
descripción detallada 325  
valores y argumentos 325

DLURLSERVER, función  
descripción básica 241  
descripción detallada 326  
valores y argumentos 326

DLVALUE, función  
descripción básica 241  
descripción detallada 327  
valores y argumentos 327

DLINKTYPE, función  
descripción básica 241  
descripción detallada 321  
valores y argumentos 321

DML  
definición 58

DOUBLE  
CHAR, utilización en conversión de formato 289

DOUBLE (*continuación*)  
 precisión 90  
 rango 90  
 tipo de datos 90

DOUBLE, función  
 descripción básica 241  
 descripción detallada 329  
 valores y argumentos 329

DOUBLE, tipo de datos 850

DOUBLE o DOUBLE\_PRECISION, función  
 descripción básica 241

DOUBLE-PRECISION, tipo de datos 850

DROP, sentencia  
 descripción detallada 1005

DROP CONSTRAINT, cláusula de sentencia ALTER TABLE 587

DROP CHECK, cláusula de sentencia ALTER TABLE 586

DROP FOREIGN KEY, cláusula ALTER TABLE, sentencia 586

DROP PARTITIONING KEY, cláusula de sentencia ALTER TABLE 587

DROP PRIMARY KEY, cláusula ALTER TABLE, sentencia 586

DROP TRANSFORM 1005

DROP UNIQUE, cláusula ALTER TABLE, sentencia 586

duración  
 etiquetada 184  
 fecha y hora 185  
 formato de fecha 185  
 formato de hora 185  
 restar 187  
 sumar 187

duración etiquetada  
 descripción 184

duraciones  
 definición 184

duraciones etiquetadas  
 diagrama 184  
 valores 184

DYNAMICRULES 81

## E

ejecución  
 privilegios de paquete 1061  
 revocar privilegios de paquete 1124

ejecución remota del SQL 49

elemento de código 34

EMP\_ACT, tabla de muestra 1433

EMP\_PHOTO, tabla de muestra 1435

EMP\_RESUME, tabla de muestra 1435

EMPLOYEE, tabla de muestra 1429

ENCRYPT, función  
 descripción básica 241  
 descripción detallada 331  
 valores y argumentos 331

END DECLARE SECTION, sentencia 1034  
 descripción detallada 1034

enlace 1061  
 recuperación de datos, papel en la optimización 9  
 revocación de todos los privilegios 1124  
 semántica de función 174  
 semántica de método 174  
 sentencia enlazada 45

enlace de datos  
 INSERT, sentencia 1086

enlaces de datos  
 creación 327  
 especificaciones BNF 1521  
 extracción de la vía de acceso y del nombre de archivo 323, 324  
 extracción de tipoenlace 321  
 extracción del comentario 320  
 extracción del esquema 325  
 extracción del servidor de archivos 326  
 extracción del URL completo 322

entero-escala  
 DECIMAL, función 311

entero-precisión  
 DECIMAL, función 311  
 valores por omisión para tipos de datos 311

enteros  
 definición de constantes 129  
 ejemplo de sintaxis de constantes 129  
 en cláusula ORDER BY 523  
 resumen de conversión decimal 107

enteros binarios  
 tipos de datos 83

enteros grandes 90

errores  
 cerrar cursor 1097

errores de referencia ambigua 148

errores de referencia no definida 148

escala de datos  
 comparaciones en SQL, visión general 114  
 conversión de número en SQL 107  
 determinada por la variable SQLLEN 1271  
 en operaciones aritméticas 182

escala de números  
 determinada por la variable SQLLEN 1276

espacio de tabla  
 identificación  
 CREATE TABLE, sentencia 869  
 índice  
 CREATE TABLE, sentencia 870

espacio de tablas  
 convenios de nombre 77  
 descripción del nombre 77

espacio de tablas SMS  
 CREATE TABLESPACE, sentencia 896  
 descripción 39

espacio gestionado por base de datos 39

espacio gestionado por el sistema (SMS)  
 descripción 39

espacios  
 reglas que rigen 71

espacios de tablas  
 adición de comentarios al catálogo 628  
 agrupaciones de almacenamientos intermedios 673  
 CREATE TABLESPACE, sentencia 893  
 definición 39  
 descripción 39  
 DROP, sentencia 1018  
 otorgar privilegios 1078  
 requisitos de red denominación 1116  
 revocar privilegios 1137  
 suprimir utilizando la sentencia DROP 1005  
 tamaño de página 895

espacios de tablas DMS  
 CREATE TABLESPACE, sentencia 896



espacios de tablas DMS  
(*continuación*)  
  descripción 39

especificación de índice 58

especificaciones  
  CAST 193

esquemas  
  adición de comentarios al  
  catálogo 628  
  controlar la utilización 13  
  crear esquema implícito,  
  otorgando autorización 1057  
  crear esquema implícito,  
  revocando autorización 1119  
  CREATE SCHEMA,  
  sentencia 822  
  definición 13  
  privilegios 14  
  reservado 1447

esquemas de codificación 34

esquemas implícitos  
  GRANT (autorizaciones de base  
  de datos), sentencia 1057  
  REVOKE (autorizaciones de  
  bases de datos) sentencia 1119

estabilidad de lectura (RS) 25  
  tabla de comparación 1453

estabilidad del cursor (CS) 26  
  tabla de comparación 1453

estado cerrado del cursor 1097

estado conectado  
  descripción 52

estado de comprobación  
  pendiente 23, 1168

estado de conexión actual 53

estado de conexión inactivo 53

estado de conexión mantenido 53

estado de conexión pendiente de  
  liberación 53

estado no conectado  
  descripción 52

estados  
  conexión 53

estados de conexión  
  proceso de aplicación 51  
  unidad de trabajo distribuida 50  
  unidad de trabajo remota 45

estándares  
  establecimiento de normas para  
  el SQL dinámico 1184

estándares ISO/ANSI  
  SQLCODE 543  
  SQLSTATE 543

estáticas, sentencias de SQL  
  invocación 538

estructura de datos  
  índice, valores derivados 16

estructura SQLCA  
  visión general 542

estructuras de almacenamiento  
  agrupaciones de  
  almacenamientos  
  intermedios 39  
  ALTER BUFFERPOOL,  
  sentencia 545  
  ALTER TABLESPACE,  
  sentencia 597  
  CREATE BUFFERPOOL,  
  sentencia 673  
  CREATE TABLESPACE,  
  sentencia 893  
  descripción 39  
  espacio de tablas 39  
  grupo de nodos 39

estructuras de datos  
  columna 14  
  constantes  
  reglas para coma  
  flotante 129  
  reglas para decimales 129  
  reglas para enteros 129  
  reglas para series de  
  caracteres 129  
  reglas para series gráficas  
  (DBCS) 129  
  decimal empaquetado 1279  
  fila 14  
  numéricas 90  
  sintaxis y rango de la fecha 91  
  sintaxis y rango de la hora 91  
  valor 14  
  valores  
  fuentes 83  
  tipos de datos 83

etiqueta  
  convenios de denominación 75

etiqueta, GOTO 1225

etiquetas de sistema principal  
  GO TO, cláusula 1207

EUC (extended UNIX code)  
  consideraciones 1511

EUR  
  formato de indicación de fecha y  
  hora 92

EVENT\_MON\_STATE, función  
  descripción básica 242

EXCEPT, operador de fullselect 513

exclusivas, claves 18  
  definición 17

EXCLUSIVE, opción en sentencia  
  LOCK TABLE 1092

EXCLUSIVE MODE, conexión 653

EXECUTE, sentencia  
  descripción detallada 1036  
  uso incorporado 539

EXECUTE, sentencias  
  SQL dinámico 9

EXECUTE IMMEDIATE, sentencia  
  descripción detallada 1042  
  uso incorporado 539

EXECUTE IMMEDIATE, sentencias  
  SQL dinámico 9

EXP, función  
  descripción básica 242  
  descripción detallada 335  
  valores y argumentos 335

EXPLAIN, sentencia  
  descripción detallada 1045

EXPLAIN\_ARGUMENT, tabla  
  creación 1484  
  descripción detallada 1460

EXPLAIN\_INSTANCE, tabla  
  creación 1485  
  descripción detallada 1464

EXPLAIN\_OBJECT, tabla  
  creación 1486  
  descripción detallada 1467

EXPLAIN\_OPERATOR, tabla  
  creación 1487  
  descripción detallada 1469

EXPLAIN\_PREDICATE tabla  
  creación 1488  
  descripción detallada 1472

EXPLAIN\_STATEMENT, tabla  
  creación 1489  
  descripción detallada 1474

EXPLAIN\_STREAM, tabla  
  creación 1490  
  descripción detallada 1477

expresión-agrupación 487

expresión CASE  
  definición 190

expresión de resultado de CASE  
  tipo de datos de resultado 120

expresión de tabla común  
  descripción 27

expresión-nextval 207

expresión-prevval 207

expresión-ref-ámbito  
  operación de desreferencia 196

expresiones  
  CASE 190  
  en cláusula ORDER BY 523  
  en una subselección 474

- expresiones (*continuación*)
  - escalar, selección completa 184
  - especificaciones CAST 193
  - expresiones CAST 193
  - expresiones-agrupación en GROUP BY 487
  - formato y reglas 176
  - funciones OLAP 197
  - invocación de métodos 204
  - operaciones de
    - desreferencia 196
  - operadores aritméticos 180
  - operadores de
    - concatenación 177
  - operadores de sustitución 176
  - operadores matemáticos 176
  - operandos de coma flotante 183
  - operandos de fecha y hora 184
  - operandos decimales 182
  - operandos enteros 182
  - prioridad de operación 190
  - secuencias 207
  - SELECT, diagrama de sintaxis en la cláusula 473
  - series 177
  - sin operadores 177
  - tratamiento de los subtipos 206
  - valores 176
- expresiones de tabla
  - descripción 27
  - expresiones de tabla comunes 519
- expresiones de tabla anidadas 480
- expresiones de tabla comunes
  - definición 519
  - recursiva 520
  - sentencia select 519
- expresiones de tabla comunes recursivas 520
- EXTEND USING, cláusula
  - CREATE INDEX, sentencia 778
- extracción de comentario del valor DATALINK
  - DLCOMMENT, función 320
- extracción de esquema del valor DATALINK
  - DLURLSCHEME, función 325
- extracción de la vía de acceso y del nombre de archivo del valor DATALINK
  - DLURLPATH, función 323
  - DLURLPATHONLY, función 324
- extracción de tipoenlace del valor DATALINK
  - DLLINKTYPE, función 321
- extracción del servidor de archivos del valor DATALINK
  - DLURLSERVER, función 326
- extracción del URL completo del valor DATALINK
  - DLURLCOMPLETE, función 322
- F**
- fecha
  - CHAR, utilización en conversión de formato 289
  - duraciones de día, buscar en el rango 309
  - formato de duración 185
  - formato de serie 92
  - mes, devolución del valor de indicación de fecha y hora 373
  - utilización de año en expresiones 456
- fecha y hora
  - de GENERATE\_UNIQUE 339
  - definición de datos 91
  - duración 185
  - EUR 92
  - formato de representación de serie 94
  - función escalar VARCHAR 450
  - ISO 92
  - JIS 92
  - límites 1255
  - LOCAL 92
  - operaciones aritméticas 185, 189
  - restricción de serie de caracteres de múltiples bytes (MBCS) 94
  - tipo de datos 185
  - tipos de datos 83
    - descripción 91
  - representación de serie 92
  - USA 92
- FETCH, sentencia
  - descripción detallada 1050
  - requisitos previos del cursor para la ejecución 1050
- fila
  - cursor, efecto de cierre en FETCH 626
- fila del total 492
- fila fantasma 25, 1454
- filas
  - actualización de los valores de columna, sentencia UPDATE 1193
- filas (*continuación*)
  - asignación de valores a la variable del lenguaje principal, SELECT INTO 1145
  - asignación de valores a la variable del lenguaje principal, VALUES INTO 1205
  - autorreferente 20
  - bloqueos, efecto sobre el cursor de WITH HOLD 977
  - bloqueos en datos de filas, sentencia INSERT 1090
  - cláusula HAVING, resultados de búsqueda 495
  - cláusula SELECT, diagrama de sintaxis 473
  - claves de índice con cláusula UNIQUE 775
  - condiciones de búsqueda, sintaxis 232
  - COUNT, función 261
  - COUNT\_BIG, función 263
  - cursor, ubicación en la tabla resultante 977
  - cursor en sentencia FETCH 1097
  - definición 14
  - dependiente 20
  - descendiente 20
  - GROUP BY, utilización para limitar la cláusula SELECT 475
  - HAVING, utilización para limitar la cláusula SELECT 475
  - índices 774
  - inserción de valores, sentencia INSERT 1084
  - inserción en tablas o vistas 1082
  - insertar valores 1085
  - otorgar privilegio de exportación de datos de fila 1073
  - otorgar privilegio de importación de valores 1072
  - otorgar privilegio de inserción 1072
  - otorgar privilegio de recuperación de datos de fila 1073
  - otorgar privilegio de supresión 1072
  - padre 20
  - petición FETCH, selección de fila de cursor 977
  - restricciones que conducen a anomalías 1085
  - suprimir 991



- filas (*continuación*)
  - tablas de resultado de cláusula GROUP BY 487
- filas de subtotales 490
- filas de superagregados 491
- filas de tabulación cruzada 491
- filas superagregadas simétricas 491
- FLOAT
  - tipo de datos 90
- FLOAT, función
  - descripción básica 242
  - descripción detallada 336
  - valores y argumentos 336
- FLOAT, tipo de datos 850
- FLOOR, función
  - descripción básica 242
  - descripción detallada 337
  - valores y argumentos 337
- FOR, sentencia 1221
- FOR BIT DATA, cláusula
  - CREATE TABLE, sentencia 851
- FOR FETCH ONLY, cláusula
  - sentencia select 527
- FOR READ ONLY, cláusula
  - sentencia select 527
- foráneas, claves 19
  - definición 17
- FOREIGN KEY, cláusula
  - CASCADE, resumen de cláusula de propagación 877
  - CREATE TABLE, sentencia 875
  - nombre de restricción, convenios para 875
  - regla de supresión, convenios para 877
  - RESTRICT, cláusula de prohibición 877
  - SET NULL, operación de la cláusula 877
  - vías de acceso múltiples, consecuencias de utilización de 877
- formato de fecha European (EUR) 92
- formato de fecha International Standards Organization (ISO) 92
- formato de fecha Japanese Industrial Standard (JIS) 92
- formato de fecha USA 92
- formato de hora European (EUR) 93
- formato de hora International Standards Organization (ISO) 93
- formato de hora Japanese Industrial Standard (JIS) 93
- formato de hora LOCAL 93
- formato de hora USA 93
- formato de indicación de fecha y hora 92
- formato de indicación de fecha y hora LOCAL 92
- fragmentos en la función SUBSTR, aviso 428
- FREE LOCATOR, sentencia
  - descripción detallada 1055
- FROM, cláusula
  - DELETE, sentencia 992
  - nombres expuestos explicados 145
  - nombres no expuestos explicados 145
  - PREPARE, sentencia 1100
  - sintaxis de subselección 478
  - uso de nombres de correlación 145
- fuentes de datos en sistemas federados
  - consultar con paso a través 1423
- función
  - escalar
    - DAYOFWEEK 306
    - DAYOFWEEK\_ISO 307
    - LCASE o LOWER 356
  - función COVARIANCE o COVAR
    - descripción básica 239
  - función CHAR(SYSFUN.CHAR) 238
  - función de columna
    - argumentos 236
  - función definida por el usuario
    - CREATE FUNCTION (SQL, escalar, de tabla o de fila), sentencia 761
    - sentencia CREATE FUNCTION 695
    - sentencia CREATE FUNCTION (Tabla externa OLE DB) 741
  - función Deref
    - descripción básica 240
    - descripción detallada 317
    - tipos de referencia 317
    - valores y argumentos 317
  - función INTEGER
    - descripción detallada 353
    - valores y argumentos 353
  - función LCASE o LOWER
    - descripción de formato detallado 356
    - valores y argumentos, reglas para 356
  - función
    - LCASE(SYSFUN.LCASE) 243
  - función
    - LTRIM(SYSFUN.LTRIM) 244
  - función REGR\_AVGX 247
  - función REGR\_AVGY 247
  - función REGR\_COUNT
    - descripción básica 248
  - función REGR\_INTERCEPT o REGR\_ICPT
    - descripción básica 248
  - función REGR\_R2
    - descripción básica 248
  - función REGR\_SLOPE
    - descripción básica 248
  - función REGR\_SXX 248
  - función REGR\_SXY 248
  - función REGR\_SYY 248
  - función RTRIM
    - (SYSFUN.RTRIM) 249
  - función SQLCACHE\_SNAPSHOT
    - descripción 464
    - descripción básica 250
  - función TRUNCATE o TRUNC
    - descripción detallada 444
    - valores y argumentos 444
  - función TYPE\_ID
    - descripción básica 252
    - descripción detallada 445
    - tipos de datos 445
    - valores y argumentos 445
  - función TYPE\_SCHEMA
    - descripción básica 252
    - descripción detallada 447
    - tipos de datos 447
    - valores y argumentos 447
  - función
    - UCASE(SYSFUN.UCASE) 252
  - funciones 159, 235
    - adición de comentarios al catálogo 628
    - anidación 279
    - argumentos 235
    - columna 257
      - AVG 237, 258
      - CORR 260
      - CORRELATION 260
      - CORRELATION o CORR 238
      - COUNT 238, 261
      - COUNT\_BIG 239, 263
      - COVAR 265
      - COVARIANCE 265
      - COVARIANCE o COVAR 239

funciones (continuación)  
columna (continuación)  
funciones de regresión 272  
MAX 244, 268  
MIN 245, 270  
REGR\_AVGX 247, 272  
REGR\_AVGY 247, 272  
REGR\_COUNT 248, 272  
REGR\_ICPT 272  
REGR\_INTERCEPT 272  
REGR\_INTERCEPT OR  
REGR\_ICPT 248  
REGR\_R2 248, 272  
REGR\_SLOPE 248, 272  
REGR\_SXX 248, 272  
REGR\_SXY 248, 272  
REGR\_SYY 248, 272  
STDDEV 250, 276  
SUM 250, 277  
VAR, opciones 278  
VAR, resultados 278  
VARIANCE, opciones 278  
VARIANCE, resultados 278  
VARIANCE o VAR 253  
de SQL  
descripción 160  
definida por el usuario 159  
derivada  
descripción 160  
descripción 159, 235  
descripción del nombre 74  
en expresiones 235  
escalar  
ABS 280  
ABS o ABSVAL 236  
ABSVAL 280  
ACOS 236, 281  
ASCII 237, 282  
ASIN 237, 283  
ATAN 237, 284  
ATAN2 237, 285  
AVG 258  
BIGINT 237, 286  
BLOB 237, 287  
CEIL 288  
CEIL o CEILING 237  
CEILING 288  
CLOB 238, 296  
COALESCE 238, 297  
CONCAT 298  
CONCAT o || 238  
COS 238, 299  
COT 238, 300  
CHAR 238, 289

funciones (continuación)  
escalar (continuación)  
CHAR (esquema  
SYSFUN) 238  
CHR 238, 295  
DATE 239, 301  
DAY 239, 303  
DAYNAME 239, 305  
DAYOFWEEK 239  
DAYOFWEEK\_ISO 239  
DAYOFYEAR 239, 308  
DAYS 240, 309  
DBCLOB 240, 310  
DECIMAL 311  
DECIMAL o DEC 240  
DECRYPT\_BIN 240  
DECRYPT\_CHAR 240  
definición 279  
definida por el usuario 469  
DEGREES 240, 316  
DEREF 240, 317  
DIFFERENCE 240, 318  
DIGITS 241, 319  
DLCOMMENT 241, 320  
DLURLCOMPLETE 241, 322  
DLURLPATH 241, 323  
DLURLPATHONLY 241, 324  
DLURLSCHEME 241, 325  
DLURLSERVER 241, 326  
DLVALUE 241, 327  
DLLINKTYPE 241, 321  
DOUBLE 241, 329  
DOUBLE o  
DOUBLE\_PRECISION 241  
DOUBLE\_PRECISION 329  
ENCRYPT 241, 331  
EVENT\_MON\_STATE 242,  
334  
EXP 242, 335  
FLOAT 242, 336  
FLOOR 242, 337  
GENERATE\_UNIQUE 242,  
339  
GET\_ROUTINE\_SAR 242,  
247  
GETHINT 242, 338  
GRAPHIC 242, 341  
GROUPING 242, 266  
HEX 242, 342  
HOUR 243, 344  
IDENTITY\_VAL\_LOCAL 243,  
345  
INSERT 243, 351  
INTEGER 353  
INTEGER o INT 243

funciones (continuación)  
escalar (continuación)  
JULIAN\_DAY 243, 355  
LCASE 243, 357  
LCASE (esquema  
SYSFUN) 243  
LEFT 243, 358  
LENGTH 244, 359  
LN 244, 361  
LOCATE 244, 362  
LOG 244, 363  
LOG10 244, 364  
LONG\_VARCHAR 244, 365  
LONG\_VARGRAPHIC 244,  
366  
LTRIM 244, 367, 368  
LTRIM (esquema  
SYSFUN) 244  
MICROSECOND 245, 369  
MIDNIGHT\_SECONDS 245,  
370  
MINUTE 245, 371  
MOD 245, 372  
MONTH 245, 373  
MONTHNAME 245, 374  
MQPUBLISH 246, 375  
MQREAD 246, 378  
MQRECEIVE 246, 380  
MQSEND 246, 382  
MQSUBSCRIBE 246, 384  
MQUNSUBSCRIBE 246, 386  
MULTIPLY\_ALT 246, 388  
NODENUMBER 246, 390  
NULLIF 246, 392  
PARTITION 246, 393  
POSSTR 246, 395  
POWER 247, 398, 400  
QUARTER 247, 399  
RADIANS 247  
RAISE\_ERROR 247, 401  
RAND 247, 403  
REAL 247, 404  
REC2XML 247, 405  
REPEAT 248, 411  
REPLACE 248, 412  
restricciones, visión  
general 279  
RIGHT 248, 413  
ROUND 249, 414  
RTRIM 249, 416, 417  
RTRIM (esquema  
SYSFUN) 249  
SECOND 249, 418  
SIGN 249, 419  
SIN 249, 420

- funciones (*continuación*)
    - escalar (*continuación*)
      - SMALLINT 249, 421
      - SOUNDEX 250, 422
      - SPACE 250, 423
      - SQRT 250, 424
      - SUBSTR 250, 425
      - TABLE\_NAME 250, 429
      - TABLE\_SCHEMA 250, 431
      - TAN 250, 434
      - TIME 251, 435
      - TIMESTAMP 251, 436
      - TIMESTAMP\_ISO 251, 438
      - TIMESTAMPDIFF 251, 439
      - TRANSLATE 252, 441
      - TRUNC 444
      - TRUNC o TRUNCATE 252
      - TRUNCATE 444
      - TYPE\_ID 252, 445
      - TYPE\_NAME 252, 446
      - TYPE\_SCHEMA 252, 447
      - UCASE 252, 448
      - UCASE (esquema SYSFUN) 252
      - UPPER 448
      - VALUE 252, 449
      - VARCHAR 253, 450
      - VARGRAPHIC 253, 452
      - WEEK 253, 454
      - WEEK\_ISO 253, 455
      - YEAR 253, 456
    - escalares
      - DECRYPT\_BIN 314
      - DECRYPT\_CHAR 314
    - externa
      - descripción 160
    - incorporada 159
    - OLAP
      - DENSE\_RANK 197
      - DENSERANK 197
      - RANK 197
      - ROW\_NUMBER 197
      - ROWNUMBER 197
    - procedimientos 465
    - resolución 162
    - signatura 161
    - tabla 457
      - MQREADALL 246, 458
      - MQRECEIVEALL 246, 461
      - SQLCACHE\_SNAPSHOT 250, 464
    - transformación
      - sentencia CREATE
        - TRANSFORM, sintaxis 903
    - vía de acceso de SQL 161
  - funciones de regresión
    - descripción detallada 272
    - REGR\_AVGX 272
    - REGR\_AVGY 272
    - REGR\_COUNT 272
    - REGR\_ICPT 272
    - REGR\_INTERCEPT 272
    - REGR\_R2 272
    - REGR\_SLOPE 272
    - REGR\_SXX 272
    - REGR\_SXY 272
    - REGR\_SYY 272
  - funciones de SQL
    - descripción 160
  - funciones de transformación
    - CREATE TRANSFORM, sentencia
      - sintaxis 903
  - funciones definidas por el usuario (UDF)
    - definición 469
    - descripción 159
    - DROP, sentencia 1005
    - GRANT (autorizaciones de base de datos), sentencia 1057
    - REVOKE (autorizaciones de bases de datos) sentencia 1119
  - funciones derivadas
    - descripción 160
  - funciones escalares
    - argumentos 236
    - descripción 279
  - funciones externas
    - descripción 160
  - funciones incorporadas
    - descripción 159, 235
  - FUNCTION, cláusula
    - COMMENT ON, sentencia 630
- G**
- generación
    - valores exclusivos 339
  - generación aleatoria en claves de particionamiento 872
  - generación de errores
    - RAISE\_ERROR, función 401
  - GENERATE\_UNIQUE, función
    - descripción básica 242
    - descripción detallada 339
  - gestor de bases de datos
    - autorización de creación
      - DBADM, otorgar 1057
    - base de datos relacional
      - distribuida 43
    - cargar la base de datos,
      - autorización para 1057
  - gestor de bases de datos (*continuación*)
    - conmutación de tareas, sentencia
      - COMMIT 640
    - control, otorgar autorización,
      - sentencia de SQL para 1056
    - guardar cambios, sentencia
      - COMMIT 640
    - límites 1253
    - SQL, interpretación de 9
    - vistas de catálogo 32
  - gestor de condiciones
    - declarar 1236
  - gestores de condiciones
    - declarar 1236
  - GET DIAGNOSTICS,
    - sentencia 1223
  - GET\_ROUTINE\_SAR, función
    - descripción básica 242
  - GETHINT, función
    - descripción básica 242
    - descripción detallada 338
    - valores y argumentos 338
  - glosario
    - definiciones 1525
    - términos 1525
  - GO TO, cláusula
    - WHENEVER, sentencia 1207
  - GOTO, sentencia 1225
  - Gran objeto de caracteres 85
  - grandes objetos binarios (BLOB) 85
    - descripción de función
      - escalar 287
    - serie 85
  - grandes objetos de caracteres de doble byte (DBCLOB)
    - definición 85
    - serie 85
  - GRANT
    - autorización de base de datos
      - descripción detallada 1056
      - CREATE ON SCHEMA 1064
      - CREATE ON SEQUENCE 1067
    - GRANT (privilegios de esquema),
      - sentencia
        - descripción detallada 1064
    - GRANT (privilegios de secuencia),
      - sentencia
        - descripción detallada 1067
    - GRANT (Privilegios de servidor)
      - descripción detallada 1068
    - GRANT (privilegios para espacios de tablas),
      - sentencia
        - descripción detallada 1078

- GRANT, sentencia
  - CONTROL ON INDEX
    - descripción detallada 1059
  - nombre de autorización 79
  - nombre de autorización, utilización en 81
  - privilegios de apodo, vista o tabla
    - descripción detallada 1070
  - Privilegios de paquete
    - descripción detallada 1061
  - privilegios para apodos 1070
  - privilegios para tablas 1070
  - privilegios para vistas 1070
- GRAPHIC, función
  - descripción básica 242
  - descripción detallada 341
  - valores y argumentos 341
- GRAPHIC, tipo de datos
  - para CREATE TABLE 852
- GROUP BY, cláusula
  - de subselección, reglas y sintaxis 487
  - resultados de subselección 475
- GROUPING, función 266
  - descripción básica 242
- grupos de nodos
  - adición de comentarios al catálogo 628
  - adición de nodos 552
  - adición de particiones 552
  - convenios de denominación 75
  - creación 801
  - creación de mapas de particionamiento 802
  - descripción 39
  - eliminación de un nodo 552
  - eliminación de una partición 552
  - nombre, sintaxis 75
- H**
- HAVING, cláusula
  - condiciones de búsqueda con subselección 495
  - resultados de subselección 475
- HEX, función
  - descripción básica 242
  - descripción detallada 342
  - valores y argumentos 342
- hora
  - CHAR, utilización en conversión de formato 289
  - devolución de la indicación de la fecha y hora de valores 436
  - hora (*continuación*)
    - devolución de segundos del valor de indicación de fecha y hora 418
    - devolución de valores basados en la hora 435
    - formato de duración 185
    - indicaciones de la fecha y hora
      - longitud de serie 91
      - representación interna 91
    - microsegundo, devolución del valor de indicación de fecha y hora 369
    - minuto, devolución del valor de indicación de fecha y hora 371
    - operaciones aritméticas, reglas 188
    - series 93
    - tipos de datos 83
    - utilización de la hora en una expresión 435
    - valores de hora, utilización en una expresión (HOUR) 344
  - HOUR, función
    - descripción básica 243
    - descripción detallada 344
    - valores y argumentos 344
  - HTML
    - programas de ejemplo 1638
- I**
- ID de autorización
  - descripción 79
- ID de autorización de ejecución 80
- ID de autorización durante la ejecución 82
- ID de autorización en sentencias dinámicas, visión general de 81
- identificador de idioma
  - manuales 1638
- identificador de objeto (OID) 868
  - CREATE TABLE, sentencia 868
  - CREATE VIEW, sentencia 961
- identificadores
  - límites de longitud 1253
  - SQL, descripción 71
  - SQL, ordinario 72
  - SQL, sistema principal 71
- identificadores de sistema principal
  - definición 72
  - en variable del lenguaje principal 152
  - en variables del lenguaje principal 75
  - sentencia de SQL 71
- identificadores de SQL sensibles a mayúsculas/minúsculas 71
- identificadores delimitados, SQL 72
- identificadores SQL
  - base de datos 72
- IDENTITY, columnas
  - CREATE TABLE, sentencia 864
- IDENTITY\_VAL\_LOCAL, función
  - descripción básica 243
  - descripción detallada 345
  - valores y argumentos 345
- IF, sentencia 1227
- IMMEDIATE, palabra clave
  - EXECUTE IMMEDIATE, sentencia 1042
- implícita, conexión
  - sentencia CONNECT 653
- imprimir manuales PDF 1640
- IN EXCLUSIVE MODE, cláusula en sentencia LOCK TABLE 1092
- IN SHARE MODE, cláusula, sentencia LOCK TABLE 1092
- IN\_TRAY, tabla de muestra 1436
- INCLUDE, cláusula
  - CREATE INDEX, sentencia 777
- INCLUDE, sentencia
  - descripción detallada 1080
- incorporación de sentencias de SQL procedimientos SQL 539
- incrementar un hora, reglas 189
- incrementar una fecha, reglas 187
- INDEX, cláusula
  - CREATE INDEX, sentencia 774, 776
  - sentencia GRANT (tabla, vista o apodo) 1072
  - sentencia REVOKE, eliminar privilegios 1132
- INDEX, cláusula de la sentencia COMMENT 632
- INDEX, palabra clave
  - DROP, sentencia 1011
- índice
  - adición de comentarios al catálogo 628
  - correspondencia con los valores de fila insertados 1085
  - definición 16
  - relaciones de vista 16
  - usos 16
- índice, asistente de 1647
- índices
  - clave de unicidad, utilización en coincidencias 579

- índices (*continuación*)
  - clave primaria, utilización en coincidencia 580
  - ID de autorización en nombre 79
  - otorgar control 1059, 1072
  - revocar privilegios 1122
  - suprimir utilizando la sentencia DROP 1005
- información de última hora 1639
- información en línea
  - buscar 1648
  - visualizar 1644
- iniciar
  - unidad de trabajo nueva 1139
- inserción en almacenamiento intermedio 1087
- INSERT
  - insertar valores 1085
  - restricciones que conducen a anomalías 1085
- INSERT, cláusula
  - sentencia GRANT (tabla, vista o apodo) 1072
  - sentencia REVOKE, eliminar privilegios 1132
- INSERT, función
  - descripción básica 243
  - descripción detallada 351
  - valores y argumentos 351
- INSERT, sentencia
  - descripción detallada 1082
- instalación
  - navegador Netscape 1644
- INTEGER
  - descripción 90
  - precisión 90
  - rango 90
  - tipo de datos 90
- INTEGER, tipo de datos 850
- INTEGER o INT, función
  - descripción básica 243
- integridad de los datos
  - ejemplo de punto de coherencia 29
  - proteger utilizando bloqueos 1092
- integridad de referencia 19, 20
- interfaz de nivel de llamada (CLI)
  - definición 31
- INTERSECT, operador
  - de selección completa, papel en la comparación 513
  - filas duplicadas, utilización de ALL 513
- INTO, cláusula
  - FETCH, sentencia de sustitución de variables del lenguaje principal 1051
  - FETCH, uso de la sentencia en variable lenguaje principal 151
- INSERT, sentencia, denominación de tabla o vista 1083
- PREPARE, sentencia 1099
- restricciones en la utilización 1083
- SELECT INTO, sentencia 1145
- SELECT INTO, uso de la sentencia en variable lenguaje principal 151
- sentencia DESCRIBE, nombre de área SQLDA 997
- valores de programas de aplicación 151
- VALUES INTO, sentencia 1205
- invocación de métodos 204
- invocación de secuencia 207
- IS, cláusula
  - sentencia COMMENT 637
- ISO
  - formato de indicación de fecha y hora 92
- ITERATE, sentencia 1229
- J**
- JDBC
  - programas 32
- JIS
  - formato de indicación de fecha y hora 92
- juego de caracteres ampliado 70
- juego de caracteres de múltiples bytes (MBCS) 70
- juego de caracteres de un solo byte (SBCS) 70, 88
- juegos de caracteres
  - definición 34
- JULIAN\_DAY, función
  - descripción básica 243
  - descripción detallada 355
  - valores y argumentos 355
- L**
- LCASE, función
  - descripción básica 243
  - descripción detallada 357
  - valores y argumentos 357
- LEAVE, sentencia 1230
- lectura no confirmada (UR) 26
- lectura no repetible 1454
- lectura repetible (RR) 24
  - tabla de comparación 1453
- lectura sucia 1454
- LEFT, función
  - descripción básica 243
  - descripción detallada 358
  - valores y argumentos 358
- LENGTH, función
  - descripción básica 244
  - descripción detallada 359
  - valores y argumentos 359
- Lenguaje de consulta estructurada (SQL)
  - asignaciones 105
  - caracteres, rango de 69
  - comentarios, reglas 69
  - constantes, definición 129
  - espacios, definición 69
  - identificadores, definición
    - identificadores delimitados, descripción 71
    - identificadores ordinarios, descripción 71
  - juego de caracteres de doble byte (DBCS), consideraciones 69
  - nombres de variable utilizados 72
  - números 90
  - operación de comparación, visión general 105
  - operandos básicos, asignaciones y comparaciones 105
  - series de caracteres, visión general 87
  - símbolos 70
  - símbolos, definición
    - símbolos delimitadores 69
    - símbolos ordinarios 69
  - valores
    - fuentes 83
    - tipos de datos 83
- lenguaje de definición de datos (DDL)
  - definición 9, 58
- lenguaje de manipulación de datos (DML)
  - definición 58
- letras
  - rango 70
- LIKE, predicado
  - reglas 223
- límites
  - fecha y hora 1255

- límites (*continuación*)
    - gestor de bases de datos 1256, 1260
    - longitud de identificador numérico 1253
    - serie 1255
    - SQL 1253
  - límites del gestor de bases de datos 1256
  - límites específicos de tamaño de página del gestor de bases de datos 1260
  - lista de selección
    - descripción 473
    - reglas de aplicación y sintaxis 475
    - reglas de notación y convenciones 473
  - literales
    - visión general 129
  - LN, función
    - descripción básica 244
    - descripción detallada 361
    - valores y argumentos 361
  - LOAD, parámetro de la sentencia GRANT...ON DATABASE 1057
  - LOB
    - localizador, definición 85
    - serie, definición 85
  - LOCAL
    - formato de indicación de fecha y hora 92
  - localizador
    - definición 85
  - localizador de grandes objetos (LOB), definición 85
  - localizadores
    - descripción de variable 155
    - FREE LOCATOR, sentencia 1055
  - LOCATE, función
    - descripción básica 244
    - descripción detallada 362
    - valores y argumentos 362
  - LOCATORS 1216
  - LOCK TABLE, sentencia
    - descripción detallada 1092
  - LOG, función
    - descripción básica 244
    - descripción detallada 363
    - valores y argumentos 363
  - LOG10, función
    - descripción básica 244
    - descripción detallada 364
    - valores y argumentos 364
  - lógica de evaluación verdadera
    - reglas de condiciones de búsqueda 232
  - LONG\_VARCHAR, función
    - descripción básica 244
    - descripción detallada 365
    - valores y argumentos 365
  - LONG VARCHAR, tipo de datos para CREATE TABLE 851
  - LONG\_VARGRAPHIC, función
    - descripción básica 244
    - descripción detallada 366
    - valores y argumentos 366
  - longitudes de las expresiones LENGTH, función 359
  - LOOP, sentencia 1231
  - LTRIM, función
    - descripción básica 244
    - descripción detallada 367, 368
    - valores y argumentos 367, 368
- ## M
- MANAGED BY, cláusula de la sentencia CREATE TABLESPACE 893
  - manuales 1629, 1640
  - marcadores de parámetros
    - con tipo 1101
    - en expresiones, predicados y funciones 1101
    - especificación CAST 194
    - OPEN, sentencia 1095
    - PREPARE, sentencia 1101
    - reglas 1101
    - sentencia EXECUTE 1036
    - sin tipo 1101
    - sustitución en sentencia OPEN 1094
    - variables de lenguaje principal en SQL dinámico 152
  - MAX, función
    - descripción básica 244
    - descripción de formato detallado 268
    - valores y argumentos 268
  - mayúsculas, conversión a 71
  - mensajes de error
    - códigos de retorno 542
    - ejecución de
      - desencadenantes 918
      - FETCH, sentencia 1052
      - UPDATE, sentencia 1199
  - METHOD, cláusula
    - DROP, sentencia 1011
  - método definido por el usuario
    - descripción 168
  - métodos
    - convenios de denominación 75
    - definido por el usuario 168
    - descripción 168
    - invocación 204
    - nombre, sintaxis 75
  - MICROSECOND, función
    - descripción básica 245
    - descripción detallada 369
    - valores y argumentos 369
  - MIDNIGHT\_SECONDS, función
    - descripción básica 245
    - descripción detallada 370
    - valores y argumentos 370
  - MIN, función
    - descripción básica 245
    - descripción de formato detallado 270
    - valores y argumentos 270
  - MINUTE, función
    - descripción básica 245
    - descripción detallada 371
    - valores y argumentos 371
  - MOD, función
    - descripción básica 245
    - descripción detallada 372
    - valores y argumentos 372
  - MODE, palabra clave en sentencia LOCK TABLE 1092
  - módulo de reiniciador 59
  - MONTH, función
    - descripción básica 245
    - descripción detallada 373
    - valores y argumentos 373
  - MONTHNAME, función
    - descripción básica 245
    - descripción detallada 374
    - valores y argumentos 374
  - MQPUBLISH, función
    - descripción básica 246
    - descripción detallada 375
    - valores y argumentos 375
  - MQREAD, función
    - descripción básica 246
    - descripción detallada 378
    - valores y argumentos 378
  - MQREADALL, función
    - descripción básica 246
    - descripción detallada 458
    - valores y argumentos 458
  - MQRECEIVE, función
    - descripción básica 246
    - descripción detallada 380



- MQRECEIVE, función (*continuación*)  
valores y argumentos 380
- MQRECEIVEALL, función  
descripción básica 246  
descripción detallada 461  
valores y argumentos 461
- MQSEND, función  
descripción básica 246  
descripción detallada 382  
valores y argumentos 382
- MQSUBSCRIBE, función  
descripción básica 246  
descripción detallada 384  
valores y argumentos 384
- MQUNSUBSCRIBE, función  
descripción básica 246  
descripción detallada 386  
valores y argumentos 386
- MULTIPLY\_ALT, función  
descripción básica 246  
descripción de formato detallado 388  
valores y argumentos, reglas para 388
- ## N
- navegador Netscape  
instalación 1644
- NICKNAME, cláusula  
DROP, sentencia 1013
- nivel de aislamiento CS (estabilidad del cursor) 26  
tabla de comparación 1453
- nivel de aislamiento RR (lectura repetible) 24  
tabla de comparación 1453
- nivel de aislamiento RS (estabilidad de lectura) 25  
tabla de comparación 1453
- nivel de aislamiento UR (lectura no confirmada) 26  
tabla de comparación 1453
- nivel de autorización  
nombre de autorización, reglas de sintaxis para 73
- niveles de aislamiento  
comparaciones 1453  
descripción 23  
en sentencia DELETE 518, 994, 1085, 1146, 1199  
estabilidad de lectura 25  
estabilidad de lectura (RS) 1453  
estabilidad del cursor 26  
estabilidad del cursor (CS) 1453  
lectura no confirmada 26
- niveles de aislamiento (*continuación*)  
lectura no confirmada (UR) 1453  
lectura repetible 24  
lectura repetible (RR) 1453  
ninguna 1453  
tablas temporales declaradas, falta de 23
- NO ACTION, regla de supresión 875
- NODEGROUP, cláusula  
CREATE BUFFERPOOL, sentencia 674  
DROP, sentencia 1014  
sentencia COMMENT 633
- NODENUMBER, función 390  
descripción básica 246  
descripción detallada 390  
valores y argumentos 390
- nombre-atributo  
operación de desreferencia 196
- nombre-autorización  
confusión con ID de autorización 80  
uso en BIND 82  
utilización en Grant y Revoke 79  
utilización en GRANT y REVOKE 81
- nombre-columna  
INSERT, sentencia 1083
- nombre-correlación  
descripción 74  
referencia calificada 144  
reglas 144
- nombre-correlación expuesto en cláusula FROM 145
- nombre-correlación no expuesto en cláusula FROM 145
- nombre-cursor  
definición 74
- nombre-cursor, ALLOCATE 1212
- nombre de atributo  
definición 73
- nombre de autorización  
restricciones que rigen 73
- nombre de columna  
definición 74  
usos 143
- nombre de condición  
definición 74
- nombre de correlación  
cláusula FROM, reglas de subselección 478
- nombre de correlación (*continuación*)  
SELECT, diagrama de sintaxis en la cláusula 473
- nombre de índice  
denominación calificada 75  
denominación no calificada 75  
restricción de clave primaria 874  
restricción de unicidad 874
- nombre de restricción  
descripción 74
- nombre de seudónimo  
definición 73
- nombre-descriptor 74  
en sentencia FETCH 1051
- nombre-específico  
descripción 77
- nombre-esquema  
descripción 77
- nombre-sentencia  
descripción 77
- nombre-servidor  
descripción 77
- nombre-supertipo, descripción 77
- nombre-tabla  
en sentencia CREATE TABLE 843
- nombre-vista  
descripción 78  
en sentencia ALTER VIEW 613
- nombres  
identificación de columnas en subselección 474  
utilización en la supresión de filas 994
- nombres de condiciones, reglas para los 74
- nombres de correlación exclusivos  
designadores de tabla 150
- nombres de esquema  
descripción 75  
nombres reservados 1447
- NOT FOUND, cláusula  
WHENEVER, sentencia 1207
- NOT NULL  
en predicado NULL 229
- NOT NULL, cláusula  
CREATE TABLE, sentencia 855
- notas del release 1639
- null  
especificación CAST 194
- NULL  
regla de supresión SET NULL de palabra clave  
descripción 22
- NULL, reglas del predicado 229

- NULLIF, función
    - descripción básica 246
    - descripción detallada 392
    - valores y argumentos 392
  - numéricas
    - asignaciones en operaciones SQL 106
    - comparaciones 114
  - numérico
    - límites 1254
  - número decimal empaquetado, situación de la coma decimal 90
  - número decimal implícito 90
  - números de coma flotante
    - precisión 90
    - rango 90
    - tipos de datos 83
- O**
- ODBC
    - definición 31
  - OF, cláusula
    - CREATE VIEW, sentencia 960
  - OLAP
    - funciones 197
  - OLAP, funciones
    - cláusula BETWEEN 197
    - cláusula CURRENT ROW 197
    - cláusula OVER 197
    - cláusula PARTITION BY 197
    - cláusula RANGE 197
    - cláusula ROW 197
    - cláusula UNBOUNDED 197
    - ORDER BY, cláusula 197
  - ON, cláusula
    - CREATE INDEX, sentencia 776
  - ON TABLE, cláusula
    - GRANT, sentencia 1074
    - REVOKE, sentencia 1133
  - ON UPDATE, cláusula 878
  - ONLY, cláusula
    - DELETE, sentencia 993
    - UPDATE, sentencia 1195
  - opción de columna serie numérica 1413
  - opción de columna
    - sin\_blanco\_ cola\_varchar 1414
  - opción de servidor
    - collating\_sequence 1416
  - opción de servidor cpu\_ratio 1417
  - opción de servidor dbname 1417
  - opción de servidor fold\_id 1418
  - opción de servidor fold\_pw 1418
  - opción de servidor io\_ratio 1418
  - opción de servidor node 1419
  - opción de servidor password 1419
  - opción de servidor plan\_hints 1419
  - opción de servidor pushdown 1420
  - opción de servidor
    - varchar\_no\_trailing\_blanks 1420
  - opciones de columna
    - CREATE TABLE, sentencia 855
    - serie numérica 1413
    - varchar\_no\_trailing\_blanks 1414
  - opciones de servidor
    - collating\_sequence 1416
    - comm\_rate 1417
    - connectstring 1417
    - contraseña 1419
    - cpu\_ratio 1417
    - dbname 1417
    - fold\_id 1418
    - fold\_pw 1418
    - io\_ratio 1418
    - nodo 1419
    - plan\_hints 1419
    - pushdown 1420
    - varchar\_no\_trailing\_blanks 1420
  - OPEN, sentencia
    - descripción detallada 1094
  - Open database connectivity (ODBC)
    - definición 31
  - operación
    - asignaciones 105
    - fecha y hora, reglas de SQL 185
  - operación de desreferencia 196
  - expresión-ref-ámbito 196
  - operando nombre-atributo 196
  - operaciones
    - asignaciones 105, 111
    - comparaciones 105, 114, 120
    - desreferencia 196
  - operaciones de SQL
    - básicas 105
  - operador de conjunto
    - EXCEPT, comparación de diferencias 513
    - INTERSECT, papel de AND en comparaciones 513
    - tipo de datos de resultado 120
    - UNION, correspondencia con OR 513
  - operador de prefijo 181
  - operadores 181
    - aritméticos, resumen 180
  - operadores lógicos
    - reglas para condiciones de búsqueda 232
  - operandos
    - coma flotante 183
  - operandos (*continuación*)
    - decimal 182, 183
    - entero 182
    - fecha y hora
      - duración de fecha 184
      - duración de hora 184
      - duración etiquetada 184
    - reglas para decimales 182
    - reglas para enteros 182
    - series 177
  - operandos de lista
    - tipo de datos de resultado 120
  - OPTION, cláusula
    - CREATE VIEW, sentencia 964
  - OR, tabla de evaluación 232
  - orden de clasificación
    - reglas de comparación de series 115
  - orden de evaluación
    - expresiones 190
  - ORDER BY, cláusula
    - sentencia select 522
    - utilizar en funciones OLAP 197
  - ordinario, identificador de SQL 71
  - ordinarios, definición de
    - símbolos 70
  - ORG, tabla de muestra 1436
- P**
- PACKAGE, cláusula
    - DROP, sentencia 1014
    - sentencia COMMENT 633
  - padre, claves 20
  - padre, filas 20
  - padre, tablas 20
  - páginas de códigos
    - definición 34
  - palabras
    - SQL reservadas 1447, 1449
  - palabras reservadas de SQL 1449
  - paquete
    - definición 32
    - otorgar privilegios 1061
    - plan 32
    - plan de acceso 32
  - paquetes
    - adición de comentarios al catálogo 628
    - autorización para crear, otorgar 1056
    - COMMIT, efecto de la sentencia sobre el cursor 641
    - convenios de denominación 75
    - DROP FOREIGN KEY, efecto en dependientes 592



- paquetes (*continuación*)
  - DROP PRIMARY KEY, efecto en dependientes 592
  - DROP UNIQUE, efecto de la clave en dependientes 592
  - enlace, relación 45
  - ID de autorización en nombre 79
  - ID de autorización en sentencias dinámicas 81
  - ID de autorización y enlace 82
  - nombre, sintaxis 75
  - reglas al revocar privilegios 1134
  - revocación de todos los privilegios 1124
  - suprimir utilizando la sentencia DROP 1005
- parámetros
  - convenios de denominación 75
  - nombre, sintaxis 75
- paréntesis
  - prioridad de operación 190
- partición
  - bases de datos 9
- partición de datos
  - compatibilidad entre particiones 127
  - tabla de compatibilidades 128
- partición de generación aleatoria 41
- particionamiento, clave de
  - definición 17
  - finalidad 40
- particionar datos
  - desagrupación parcial 42
  - descripción 40
  - mapa de particionamiento, definición 42
  - partición de generación aleatoria 41
- PARTITION, función
  - descripción básica 246
  - descripción detallada 393
  - valores y argumentos 393
- paso a través 56
  - COMMIT, sentencia 1424
  - proceso de SQL 1423
  - restricciones 1424
  - SET PASSTHRU, sentencia 1424
- PCTFREE, cláusula
- CREATE INDEX, sentencia 778
- PDF 1640
- petionario de aplicaciones 43
- peticiones distribuidas 58
- plantillas de función
  - descripción detallada 769
- POSSTR, función
  - descripción básica 246
  - descripción detallada 395
  - valores y argumentos 395
- POWER, función
  - descripción básica 247
  - descripción detallada 398
  - valores y argumentos 398
- precisión, como atributo
  - numérico 90
- precisión de números
  - determinada por la variable SQLLEN 1276
- precisión doble, tipo de datos
  - flotante 850
- precompilación
  - inclusión de un archivo de texto externo 1080
  - iniciación y configuración de SQLDA y SQLCA 1080
  - sentencia INCLUDE, desencadenante 1080
- precompilador
  - sentencias de SQL no ejecutables 539
  - SQL estático, utilización en llamadas al Servicio en tiempo de ejecución 30
- predicado
  - básico, diagrama detallado 213
  - cuantificado 214
  - descripción 212
  - EXISTS 219
  - LIKE 223
- predicado básico
  - formato detallado 213
- predicado BETWEEN
  - diagrama detallado 217
- predicado cuantificado
  - reglas 214
- predicado EXISTS
  - descripción 219
- predicado IN
  - descripción 220
- predicados
  - BETWEEN, diagrama detallado 217
  - IN 220
  - NULL 229
  - TYPE 230
- PREPARE, sentencia
  - descripción detallada 1099
  - SQL dinámico 9
- PREPARE, sentencia (*continuación*)
  - uso incorporado 539
- PREPARE, sentencia de SQL
  - declarar dinámicamente 1099
  - sustitución de variable en sentencia OPEN 1094
- primarias, claves
  - definición 17
- PRIMARY KEY
  - CREATE TABLE, sentencia 859
- PRIMARY KEY, cláusula
  - ALTER TABLE, sentencia 580
  - CREATE TABLE, sentencia 874
- prioridad
  - del operador, nivel 190
  - orden de evaluación de operaciones 190
- privilegio de administración de bases de datos
  - autorización de administrador (DBADM) 11
- privilegio de administración del sistema
  - autorización de administrador (SYSADM) 11
- privilegio de control del sistema
  - descripción 12
- privilegio de mantenimiento del sistema 12
- privilegios
  - base de datos, efectos de revocación 1120, 1128
  - DBADM, ámbito 11
  - definición 10
  - índice, efectos de revocación 1123
  - paquete, efectos de la revocación 1126
  - Privilegio CONTROL 10
  - reglas al revocar paquetes 1134
  - revocar 1132
  - SYSADM, ámbito 11
  - SYSCTRL, ámbito 12
  - SYSMAINT, ámbito 12
  - tabla o vista, efectos de revocación 1135
  - visión general 10
  - vistas, efectos en cascada de la revocación 1134
- procedimiento
  - autorización para crear 804
- procedimiento, sentencia
  - compuesta 1233
- procedimiento SQL
  - DECLARE, sentencia 642, 1233

- procedimiento SQL (*continuación*)
    - gestores de condiciones 1236
    - sentencia CASE 1218
    - sentencia compuesta de procedimiento 1233
    - sentencia de asignación 1214
    - sentencia de gestor de condiciones 1236
    - sentencia dinámica compuesta 642
    - sentencia FOR 1221
    - sentencia GET
      - DIAGNOSTICS 1223
    - sentencia GOTO 1225
    - sentencia IF 1227
    - sentencia ITERATE 1229
    - sentencia LEAVE 1230
    - sentencia LOOP 1231
    - sentencia REPEAT 1240
    - sentencia RESIGNAL 1242
    - sentencia RETURN 1245
    - sentencia SIGNAL 1248
    - sentencia WHILE 1251
    - SET, sentencia 1214
    - variables 642, 1233
  - procedimientos
    - convenios de denominación 75
    - crear, sintaxis 804
    - nombre, sintaxis 75
  - procedimientos almacenados
    - CALL, sentencia 617
    - crear, sintaxis 804
  - PROCEDURE, cláusula de la sentencia COMMENT 633
  - proceso analítico en línea (online analytical processing - OLAP) 197
  - proceso de aplicación, definición de 27
  - proceso de confirmación
    - bloqueos, relación con los cambios no confirmados 27
  - programa de aplicación
    - simultaneidad 27
    - utilizaciones de SQLDA 1267
  - programas de ejemplo
    - HTML 1638
    - para varias plataformas 1638
  - PROJECT, tabla de muestra 1437
  - promocionar
    - prioridad 100
    - tipos de datos 100
  - PUBLIC, cláusula
    - GRANT, sentencia 1058, 1060, 1062, 1065, 1075
  - PUBLIC, cláusula (*continuación*)
    - REVOKE (privilegios de esquema), sentencia 1128
    - REVOKE, sentencia 1120, 1125
    - sentencia REVOKE, eliminar privilegios 1133
    - sentencia REVOKE (privilegios de índice) 1123
  - punto de salvar
    - liberar 1113
    - ROLLBACK TO SAVEPOINT 1139
  - puntos de salvar
    - convenios de denominación 76
    - nombre, sintaxis 76
  - PUT\_ROUTINE\_SAR, función
    - descripción básica 247
- ## Q
- QUARTER, función
    - descripción básica 247
    - descripción detallada 399
    - valores y argumentos 399
- ## R
- RADIANS, función
    - descripción básica 247
    - descripción detallada 400
    - valores y argumentos 400
  - RAISE\_ERROR, función
    - descripción básica 247
    - descripción detallada 401
    - valores y argumentos 401
  - RAND, función
    - descripción básica 247
    - descripción detallada 403
    - valores y argumentos 403
  - RANK
    - función OLAP 197
  - REAL
    - precisión 90
    - rango 90
    - tipo de datos 90
  - REAL, función
    - conversión de precisión simple 404
    - descripción básica 247
    - descripción detallada 404
    - valores y argumentos 404
  - REAL, tipo de datos 850
  - REC2XML, función
    - descripción básica 247
    - descripción detallada 405
    - valores y argumentos 405
  - recuperación
    - de aplicaciones 27
  - recurrencia
    - consulta 520
    - ejemplo 1499
  - REFERENCES, cláusula
    - sentencia GRANT (tabla, vista o apodo) 1073
    - sentencia REVOKE, eliminar privilegios 1132
  - referencia, ciclos 20
  - referencia correlacionada 486
  - referencia correlacionada, uso en una expresión de tabla anidada 149
  - referencia correlacionada, uso en una selección completa 149
  - referencia correlacionada, uso en una subconsulta 149
  - referencia de tabla
    - apodo 479
    - expresiones de tabla anidadas 480
    - nombre de tabla 479
    - nombre de vista 479
    - seudónimo 480
  - REFRESH TABLE, sentencia
    - descripción detallada 1110
  - REFRESH DEFERRED 1110
  - REFRESH IMMEDIATE 1110
  - registro
    - definición 132
  - registro de anotaciones
    - creación de una tabla sin registro de anotaciones inicial 872
  - registros
    - bloqueos en datos de filas, sentencia INSERT 1090
  - registros especiales
    - CLIENT ACCTNG 132
    - CLIENT APPLNAME 132
    - CLIENT USERID 133
    - CLIENT WRKSTNNAME 133
    - CURRENT DATE 134, 141
    - CURRENT DEFAULT TRANSFORM GROUP 134
    - CURRENT DEGREE 135
    - CURRENT EXPLAIN MODE 136
    - CURRENT EXPLAIN SNAPSHOT 137
    - CURRENT FUNCTION PATH 138
    - CURRENT NODE 138
    - CURRENT PATH 138

registros especiales (*continuación*)  
CURRENT QUERY  
OPTIMIZATION 139  
CURRENT REFRESH AGE 140  
CURRENT SCHEMA 140  
CURRENT SERVER 141  
CURRENT SQLID 140  
CURRENT TIME 141  
CURRENT TIMESTAMP 142  
CURRENT TIMEZONE 142  
definición 132  
interacción de los registros  
especiales de explicación 1495  
USER 143

regla de inserción  
con restricción de referencia 21

regla de supresión para restricción  
de referencia 22

regla de supresión SET DEFAULT  
descripción 22

reglas de conversión  
asignaciones 108  
comparaciones 117  
comparaciones de series 124  
operaciones que combinan  
series 124

reiniciador 57

reiniciadores  
nombres 78

relacional particionada  
base de datos 9

RELEASE (Conexión), sentencia  
descripción detallada 1111

RELEASE SAVEPOINT, sentencia  
descripción detallada 1113

RENAME TABLE, sentencia  
descripción detallada 1114

RENAME TABLESPACE, sentencia  
descripción detallada 1116

rendimiento  
recomendación de clave de  
particionamiento 882

REPEAT, función  
descripción básica 248  
descripción detallada 411  
valores y argumentos 411

REPEAT, sentencia 1240

REPLACE, función  
descripción básica 248  
descripción detallada 412  
valores y argumentos 412

representación de datos  
consideraciones 55

reservadas  
palabras SQL 1449

reservado  
calificadores 1447  
esquemas 1447  
nombres de esquema 1447  
palabras 1447  
palabras SQL 1447

RESIGNAL, sentencia 1242

restauración, asistente de 1647

restricción de comprobación  
ALTER TABLE, sentencia 575,  
581  
CREATE TABLE, sentencia 878  
INSERT, sentencia 1086

restricción de unicidad  
CREATE TABLE, sentencia 873

restricciones  
adición con ALTER TABLE 565  
adición de comentarios al  
catálogo 628  
control de tabla 18, 22  
de referencia 18, 19  
de unicidad 18  
eliminación con ALTER  
TABLE 565  
Tablas de Explain 1459

restricciones de comprobación de  
tabla  
descripción 22

restricciones de integridad  
adición de comentarios al  
catálogo 628

restricciones de referencia 19  
adición de comentarios al  
catálogo 628

restricciones de unicidad 18  
adición con ALTER TABLE 565  
ALTER TABLE, sentencia 579  
definición 17  
eliminación con ALTER  
TABLE 565

RESTRICT, regla de supresión 875  
descripción 22

RESULT\_STATUS  
sentencia GET  
DIAGNOSTICS 1223

retrotracción  
definición 28

RETURN, sentencia 1245

revocar  
privilegios en esquema 1127

REVOKE  
privilegios para apodos  
descripción detallada 1131  
privilegios para vistas  
descripción detallada 1131

REVOKE, sentencia  
Autorizaciones de bases de datos  
descripción detallada 1118  
nombre-autorización 79  
nombre-autorización, utilización  
en 81  
Privilegios de esquema  
descripción detallada 1127  
Privilegios de índice  
descripción detallada 1122  
Privilegios de paquete  
descripción detallada 1124  
Privilegios de servidor  
descripción detallada 1129  
privilegios para espacios de  
tablas  
descripción detallada 1137  
privilegios para tablas  
descripción detallada 1131

REXX  
END DECLARE SECTION,  
prohibición 1034

RIGHT, función  
descripción básica 248  
descripción detallada 413  
valores y argumentos 413

ROLLBACK  
cursor, efecto en 1140  
sentencia de SQL, instrucciones  
de uso detalladas para 1140

ROLLBACK, sentencia  
descripción detallada 1139

ROLLBACK TO SAVEPOINT,  
sentencia  
descripción detallada 1139

ROLLUP 490  
ejemplos 503

ROUND, función  
descripción básica 249  
descripción detallada 414  
valores y argumentos 414

ROW\_COUNT  
sentencia GET  
DIAGNOSTICS 1223

ROW\_NUMBER  
función OLAP 197

ROWNUMBER  
función OLAP 197

RTRIM, función  
descripción básica 249  
descripción detallada 416, 417  
valores y argumentos 416, 417

rutinas  
procedimientos 465

## S

- SALES, tabla de muestra 1438
- SAVEPOINT, sentencia
  - descripción detallada 1142
- SCOPE, cláusula
  - ALTER TABLE, sentencia 574, 582
  - ALTER VIEW, sentencia 613
  - CREATE TABLE, sentencia 858
  - CREATE VIEW, sentencia 962
  - en especificación CAST 195
- SCHEMA, cláusula
  - DROP, sentencia 1016
  - sentencia COMMENT 635
- SECOND, función
  - descripción básica 249
  - descripción detallada 418
  - valores y argumentos 418
- secuencias
  - DROP, sentencia 1005
  - expresión-nextval 207
  - expresión-prevval 207
  - invocación 207
- seguridad
  - sentencia CONNECT 660
- selección completa
  - CREATE VIEW, sentencia 963
  - ejemplos 515
  - escalar 184
  - múltiples operaciones, orden de ejecución 514
  - ORDER BY, cláusula 522
  - referencia de tabla 479
  - rol de subconsulta, condición de búsqueda 149
  - sintaxis detallada 512
- selección completa de fila
  - UPDATE, sentencia 1198
- selección de una sola fila 1145
- selección estática 540
- selecciones completas escalares
  - definición 184
- SELECT, cláusula
  - con palabra clave
    - DISTINCT 473
  - notación de lista, referencia de columna 473
  - sentencia GRANT (tabla, vista o apodo) 1073
  - sentencia REVOKE, eliminar privilegios 1133
- select, sentencia
  - ejemplos 530
- SELECT, sentencia
  - cursor, reglas de los marcadores de parámetros 978
  - definición 518
  - incorporación a procedimientos SQL 539
  - invocación, resumen del uso 538
  - invocación dinámica 540
  - invocación estática 540
  - invocación interactiva, limitaciones 541
  - selección completa, sintaxis detallada 512
  - subselección 472
  - tabla resultante, sentencia OPEN, relación con el cursor 1094
  - VALUES, cláusula 512
- SELECT INTO, sentencia
  - descripción detallada 1145
- sentencia ALTER NICKNAME
  - descripción detallada 548
- sentencia ALTER SERVER
  - descripción detallada 561
- sentencia ALTER USER MAPPING
  - descripción detallada 610
- sentencia CONNECT
  - conexión IMPLICIT, diagrama de transición de estado 47
  - conexión implícita 653
  - conexión no IMPLICIT, diagrama de transición de estado 49
  - desconexión del servidor actual 660
  - descripción 45
  - información sobre cómo establecer una nueva contraseña 660
  - información sobre servidor de aplicaciones, obtención 660
  - sin operandos, devolución de información 660
- sentencia CREATE
  - FUNCTION 695, 740, 749, 768
- sentencia CREATE FUNCTION (Fuente) 760
- sentencia CREATE FUNCTION (Tabla externa OLE DB) 741
- sentencia CREATE SERVER
  - descripción detallada 831
- sentencia CREATE TYPE MAPPING
  - descripción detallada 950
- sentencia CREATE USER MAPPING
  - descripción detallada 955
- sentencia CREATE VIEW, definición de 15
- sentencia CREATE WRAPPER
  - descripción detallada 974
- sentencia de SQL
  - ALLOCATE CURSOR 1212, 1213
  - ASSOCIATE LOCATORS 1216, 1217
  - CLOSE 626, 627
  - CREATE FUNCTION 695, 740, 749, 768
  - CREATE FUNCTION (Escalar externa) 696
  - CREATE FUNCTION (fuente o plantilla) 750
  - CREATE FUNCTION (SQL, escalar, de tabla o de fila) 761
  - CREATE FUNCTION (Tabla externa) 723
  - CREATE FUNCTION (Tabla externa OLE DB) 741
  - CREATE INDEX
  - EXTENSION 782
  - CREATE METHOD 790
  - CREATE TABLE 835, 892
  - CREATE TYPE (Estructurado) 923, 949
  - nombre de sentencia, convenios 77
  - sentencia CREATE FUNCTION (Fuente) 760
- sentencia de SQL ejecutable
  - visión general de los métodos 537
- sentencia de SQL no ejecutable
  - visión general de los métodos 537
- sentencia de SQL preparada
  - ejecución 1036
  - obtener información utilizando DESCRIBE 997
  - SQLDA proporciona información 1267
  - sustitución de variables del lenguaje principal 1036
- sentencia enlazada 45
- sentencia FLUSH EVENT MONITOR
  - descripción detallada 1054
- sentencias de SQL
  - ALTER BUFFERPOOL 545
  - ALTER NICKNAME 548
  - ALTER NODEGROUP 552
  - ALTER SEQUENCE 556
  - ALTER SERVER 561
  - ALTER TABLE 565
  - ALTER TABLESPACE 597

- sentencias de SQL (*continuación*)
- ALTER TYPE (Estructurado) 603
  - ALTER USER MAPPING 610
  - BEGIN DECLARE SECTION 615
  - CALL 617
  - COMMENT 628
  - COMMIT, sentencia 640
  - CONNECT (Tipo 1) 653
  - CONNECT (Tipo 2) 662
  - CONTINUE, respuesta a excepción 1207
  - convenios de sintaxis 3
  - CREATE ALIAS 670
  - CREATE BUFFERPOOL 673
  - CREATE DISTINCT TYPE 677
  - CREATE EVENT MONITOR 684
  - CREATE FUNCTION MAPPING 769
  - CREATE INDEX 774
  - CREATE NICKNAME 795
  - CREATE NODEGROUP 801
  - CREATE PROCEDURE 804
  - CREATE SCHEMA 822
  - CREATE SEQUENCE 826
  - CREATE SERVER 831
  - CREATE TABLESPACE 893
  - CREATE TRANSFORM 903
  - CREATE TRIGGER 910
  - CREATE TYPE MAPPING 950
  - CREATE USER MAPPING 955
  - CREATE VIEW 957
  - CREATE WRAPPER 974
  - DECLARE CURSOR 976
  - DECLARE GLOBAL TEMPORARY TABLE 982
  - DELETE 991
  - DESCRIBE 997
  - DISCONNECT 1002
  - DROP 1005
  - DROP TRANSFORM 1005
  - ejecución inmediata del SQL dinámico 9
  - END DECLARE SECTION 1034
  - entrada interactiva 541
  - EXECUTE 1036
  - EXECUTE IMMEDIATE 1042
  - EXPLAIN 1045
  - FETCH 1050
  - FLUSH EVENT MONITOR 1054
  - FREE LOCATOR 1055
  - GRANT (privilegios de esquema) 1064
- sentencias de SQL (*continuación*)
- GRANT (privilegios de secuencia) 1067
  - INCLUDE 1080
  - INSERT 1082
  - invocación 537
  - LOCK TABLE 1092
  - OPEN 1094
  - preparación y ejecución del SQL dinámico 9
  - PREPARE 1099
  - REFRESH TABLE 1110
  - RELEASE (Conexión) 1111
  - RELEASE SAVEPOINT 1113
  - RENAME TABLE 1114
  - RENAME TABLESPACE 1116
  - REVOKE (privilegios de apodo) 1131
  - REVOKE (privilegios de esquema) 1127
  - REVOKE (Privilegios de servidor) 1129
  - REVOKE (privilegios de tablas) 1131
  - REVOKE (privilegios de vistas) 1131
  - REVOKE (privilegios para espacios de tablas) 1137
  - ROLLBACK 1139
  - ROLLBACK TO SAVEPOINT 1139
  - SAVEPOINT 1142
  - SELECT INTO 1145
  - SET CONNECTION 1147
  - SET CONSTRAINTS 1168
  - SET CURRENT DEFAULT TRANSFORM GROUP 1149
  - SET CURRENT DEGREE 1151
  - SET CURRENT EXPLAIN MODE 1153
  - SET CURRENT EXPLAIN SNAPSHOT 1155
  - SET CURRENT FUNCTION PATH 1181
  - SET CURRENT PACKAGESET 1157
  - SET CURRENT PATH 1181
  - SET CURRENT QUERY OPTIMIZATION 1159
  - SET CURRENT REFRESH AGE 1162
  - SET ENCRYPTION PASSWORD 1164
  - SET EVENT MONITOR STATE 1166
- sentencias de SQL (*continuación*)
- SET INTEGRITY 1168
  - SET PASSTHRU 1179
  - SET PATH 1181
  - SET SCHEMA 1184
  - SET SERVER OPTION 1186
  - SET variable 1188
  - SQL compuesto (incorporado) 648
  - SQL dinámico, definición 9
  - SQL estático, definición 9
  - SQL interactivo, definición 9
  - UPDATE 1193
  - VALUES 1204
  - VALUES INTO 1205
  - WHENEVER 1207
  - WITH HOLD, atributo del cursor 977
- Sentencias de SQL
- ALTER VIEW 613
  - GRANT 1056
    - Privilegios de índice 1059
    - Privilegios de paquete 1061
    - privilegios para apodos 1070
    - privilegios para tablas 1070
    - privilegios para vistas 1070
  - GRANT (Privilegios de servidor) 1068
  - GRANT (privilegios para espacios de tablas) 1078
  - REVOKE (autorizaciones de bases de datos) 1118
  - REVOKE (privilegios de índice) 1122
  - REVOKE (privilegios de paquete) 1124
- sentencias de SQL activadas
- sentencia SET variable 1188
- sentencias de SQL ejecutables
- resumen de proceso 538
- sentencias de SQL incorporadas
- ejecución de series de caracteres, EXECUTE IMMEDIATE 1042
  - requisitos 538
- sentencias de SQL no ejecutables
- requisitos del precompilador 539
- sentencias explícibles
- definición 1045
- serie
- asignación reglas de conversión 108
  - BLOB 85
  - constante carácter 130

serie (*continuación*)

- constante (*continuación*)
  - hexadecimal 130
- definición 33
- límites 1255
- LOB 84

serie CLOB 85

serie de caracteres

- datos de bit, definición
- definición 88

serie de caracteres de doble byte (DBCS)

- devolución de serie 452

serie de caracteres vacía 87

serie de diagnóstico

- en la función
- RAISE\_ERROR 401

serie de sentencia

- PREPARE, sentencia
- reglas 1100

serie de sentencia, reglas para la creación 1042

series

- CLOB 85
- expresiones 177
- operandos 177

series de caracteres

- asignación 107
- BLOB, representación de
  - serie 287
- CLOB 85
- comparaciones 115
- constante hexadecimal 130
- constantes, rango y precisión 130
- datos mixtos 88
- datos SBBCS, definición 88
- definición de igualdad 115
- descripción detallada 87
- devolución de nombre de
  - variable del lenguaje principal 441
- función escalar VARCHAR 450
- función escalar
  - VARGRAPHIC 452
- igualdad, ejemplos de orden de clasificación 115
- longitud fija, descripción 87
- longitud variable, descripción 87
- operadores aritméticos, uso prohibido 180
- POSSTR, función escalar 395
- sentencia de SQL, ejecución como 1042

series de caracteres (*continuación*)

- serie de caracteres de doble
  - byte 452
- serie de sentencia de SQL, reglas para la creación 1042
- sintaxis de serie de
  - conversión 441
- tipos de datos 83
- tipos de datos de longitud
  - fija 83
  - tipos de datos de longitud variable 83
  - vacía, en comparación al valor nulo 87

series gráficas

- devolución de nombre de
  - variable del lenguaje principal 441
- sintaxis de serie de
  - conversión 441
- tipo de datos de longitud fija 88
- tipo de datos de longitud
  - variable 88
- tipos de datos de longitud
  - fija 83
  - tipos de datos de longitud variable 83

series LONG VARCHAR

- atributos 87
- restricciones 87

series LONG VARGRAPHIC

- atributos 88
- restricciones 88

series VARCHAR

- atributos 87
- restricciones 87

series VARGRAPHIC

- atributos 88
- restricciones 88

servicios en tiempo de ejecución, SQL estático 30

servidor

- otorgar privilegios 1068

servidor de aplicaciones

- descripción 43
- función en conexiones 44

servidor federado 56

SET, cláusula

- sentencia UPDATE, nombres de columna y valores 1196

SET, sentencia 1214

SET CONNECTION, sentencia

- conexión no satisfactoria, descripción detallada 1148

SET CONNECTION, sentencia (*continuación*)

- conexión satisfactoria, descripción detallada 1147
- descripción detallada 1147

SET CONSTRAINTS, sentencia

- descripción detallada 1168

SET CURRENT DEFAULT

- TRANSFORM GROUP, sentencia
- descripción detallada 1149

SET CURRENT DEGREE, sentencia

- descripción detallada 1151

SET CURRENT EXPLAIN MODE, sentencia

- descripción detallada 1153

SET CURRENT EXPLAIN

- SNAPSHOT, sentencia
- descripción detallada 1155

SET CURRENT FUNCTION PATH, sentencia

- descripción detallada 1181

SET CURRENT PACKAGESET

- descripción detallada 1157

SET CURRENT PATH, sentencia

- descripción detallada 1181

SET CURRENT QUERY

- OPTIMIZATION, sentencia
- descripción detallada 1159

SET CURRENT REFRESH AGE, sentencia

- descripción detallada 1162

SET CURRENT SQLID, sentencia

- descripción detallada 1184

SET ENCRYPTION PASSWORD, sentencia

- descripción detallada 1164

SET EVENT MONITOR STATE, sentencia

- descripción detallada 1166

SET INTEGRITY, sentencia

- descripción detallada 1168

SET NULL, regla de supresión 875

- descripción 22

SET PASSTHRU, sentencia

- descripción detallada 1179
- independencia de la sentencia COMMIT 640
- independencia de la sentencia ROLLBACK 1139
- utilización 1424

SET PATH, sentencia

- descripción detallada 1181

SET SCHEMA, sentencia

- descripción detallada 1184



- SET SERVER OPTION, sentencia
  - descripción detallada 1186
  - independencia de la sentencia COMMIT 640
  - independencia de la sentencia ROLLBACK 1139
- SET variable, sentencia
  - descripción detallada 1188
- seudónimo
  - adición de comentarios al catálogo 628
  - CREATE ALIAS, sentencia 670
  - definición 16
  - descripción 78
  - TABLE\_NAME, función 429
  - TABLE\_SCHEMA, función 431
- seudónimos
  - suprimir utilizando la sentencia DROP 1005
- SHARE, opción, sentencia LOCK TABLE 1092
- SHARE MODE, conexión 653
- SIGN, función
  - descripción básica 249
  - descripción detallada 419
  - valores y argumentos 419
- SIGNAL, sentencia 1248
- signo, como atributo numérico 90
- signo de interrogación (?)
  - marcador de parámetros EXECUTE 1036
- símbolos
  - como elemento de lenguaje 69
  - delimitadores 70
  - espacios, reglas 71
  - mayúsculas y minúsculas 71
  - ordinarios 70
- simultaneidad
  - aplicación 27
  - prevención
    - LOCK TABLE, sentencia 1092
  - tablas con el parámetro NOT LOGGED INITIALLY, restricción 872
- SIN, función
  - descripción básica 249
  - descripción detallada 420
  - valores y argumentos 420
- sinónimos
  - calificación de un nombre de columna 144
  - CREATE ALIAS, sentencia 670
  - DROP ALIAS, sentencia 1008
- sintaxis de sentencias de SQL
  - convenios de nombre de sentencia 77
  - convenios de nombre de variable SQL 77
  - convenios de
    - nombre-específico 77
  - escape, caracteres 72
  - identificadores sensibles a las mayúsculas y minúsculas, reglas 71
  - nombre-cursor, definición 74
- sintaxis de SQL
  - AVG, resultados de la función en el conjunto de columnas 258
  - comparación de dos predicados, condiciones verdaderas 213, 230
  - condiciones de búsqueda, formatos y reglas 232
  - condiciones de búsqueda de cláusula WHERE 486
  - convenios de denominación, listado, definiciones 72
  - función COUNT, argumentos y resultados 261
  - función COUNT\_BIG, argumentos y resultados 263
  - función
    - SQLCACHE\_SNAPSHOT, resultados de un conjunto de pares de números 464
  - función STDDEV, resultados 276
  - GENERATE\_UNIQUE, función 339
  - GROUP BY, utilización la cláusula en subselección 487
  - múltiples operaciones, orden de ejecución 514
  - palabra clave DISTINCT, consultas 257
  - predicado básico, diagrama detallado 213
  - predicado EXISTS 219
  - predicado IN, descripción detallada 220
  - predicado TYPE 230
  - resultados de función
    - VARIANCE 278
  - resultados de la función
    - CORRELATION 260
  - resultados de la función
    - COVARIANCE 265
  - resultados de las funciones de regresión 272
- sintaxis de SQL (*continuación*)
  - SELECT, descripción detallada de la cláusula 473
  - SELECT, métodos de invocación de la sentencia 538
  - sentencias ejecutables
    - incorporadas 539
  - sentencias no ejecutables
    - incorporadas 539
  - valor nulo, definición 84
  - valores, visión general 83
- Sintaxis SQL
  - escala de datos en SQL 90
  - fechas, descripción detallada 91
  - horas, descripción detallada 91
  - predicado BETWEEN, reglas 217
  - predicado LIKE, reglas 223
  - tipos de datos 82
- sistema federado DB2 56
  - apodo 56
  - compensación 56
  - correlación de funciones 56
  - correlación de tipos de datos 56
  - correlación de usuarios 56
  - especificación de índice 56
  - módulo de reiniciador 56
  - paso a través 56
  - peticiones distribuidas 56
  - reiniciador 56
  - servidor federado 56
- sistemas federados
  - paso a través 1423
- SMALLINT
  - descripción 90
  - precisión 90
  - rango 90
- SMALLINT, función
  - descripción básica 249
  - descripción detallada 421
  - valores y argumentos 421
- SMALLINT, tipo de datos 850
  - descripción 90
  - precisión 90
  - rango 90
- SmartGuides
  - asistentes 1646
- SOME
  - predicado cuantificado 214
- SOUNDEX, función
  - descripción básica 250
  - descripción detallada 422
  - valores y argumentos 422
- SPACE, función
  - descripción básica 250

- SPACE, función (*continuación*)
  - descripción detallada 423
  - valores y argumentos 423
- SPECIFIC FUNCTION, cláusula
  - COMMENT, sentencia 632
- SPECIFIC PROCEDURE, cláusula
  - sentencia COMMENT 634
- SQL
  - comentarios 544
  - límites 1253
  - reglas para sentencias
    - estáticas 544
- SQL compuesto (dinámico)
  - variables 642
- SQL compuesto (incorporado),
  - sentencia
    - combinación de sentencias en un
      - bloque 648
- SQL dinámico
  - características 81
  - DECLARE CURSOR,
    - sentencia 540
  - definición 9
  - descripción 31
  - EXECUTE, sentencia 539
  - FETCH, sentencia 540
  - OPEN, sentencia 540
  - PREPARE, sentencia 9, 539, 540,
    - 1099
    - utilizando DESCRIBE 997
  - sentencia EXECUTE 9
  - SQLDA utilizada con 1267
- SQL estático
  - código fuente, diferencias del
    - SQL dinámico 30
  - DECLARE CURSOR,
    - sentencia 540
  - definición 9
  - FETCH, sentencia 540
  - invocación 540
  - OPEN, sentencia 540
- SQL incorporado para Java (SQLJ)
  - programas 32
- SQL interactivo
  - CLOSE, ejemplo 30
  - DECLARE CURSOR, ejemplo 30
  - definición 9, 30
  - DESCRIBE, ejemplo 30
  - FETCH, ejemplo 30
  - OPEN, ejemplo 30
  - PREPARE, ejemplo 30
  - sentencia SELECT, ejemplo
    - dinámico 30
- SQL92
  - establecimiento de normas para
    - el SQL dinámico 1184
- SQLCA (área de comunicaciones
  - SQL)
    - descripción detallada 1261
    - entrada cambiada por
      - UPDATE 1199
- SQLCA (área de comunicaciones
  - SQL), cláusula
    - INCLUDE, sentencia 1080
- SQLCODE
  - descripción 542
  - valores de código de retorno,
    - tabla 542
- SQLDA
  - descripciones de variables del
    - lenguaje principal, sentencia
      - OPEN 1095
    - información de sentencia
      - preparada,
        - almacenamiento 1099
- SQLDA (área de descriptores del
  - SQL)
    - contenido 1267
    - FETCH, sentencia 1051
- SQLDA (área de descriptores SQL),
  - cláusula
    - INCLUDE, sentencia,
      - especificación 1080
- SQLERROR, cláusula
  - WHENEVER, sentencia 1207
- sqlstate
  - en la función
    - RAISE\_ERROR 401
- SQLSTATE
  - descripción 543
  - estándar ISO/ANSI SQL92 543
- SQLWARNING, cláusula
  - WHENEVER, sentencia 1207
- SQRT, función
  - descripción básica 250
  - descripción detallada 424
  - valores y argumentos 424
- STAFF, tabla de muestra 1439
- STAFFG, tabla de muestra 1440
- STDDEV, función
  - descripción básica 250
  - descripción detallada 276
- subconsulta
  - utilizar selección completa como
    - condición de búsqueda 149
- subconsultas
  - cláusula HAVING 495
- subconsultas (*continuación*)
  - en cláusula HAVING,
    - ejecución 495
  - en cláusula WHERE 486
- subselección
  - definición 472
  - ejemplo de secuencia de
    - operaciones 472
  - ejemplos 496
  - FROM, relación de la cláusula
    - con subselección 472
- subseries
  - inicio, establecimiento 425
  - longitud, definición 425
  - precauciones y restricciones 428
  - ubicación en la serie 425
- SUBSTR, función
  - descripción básica 250
  - descripción detallada 425
  - valores y argumentos 425
- subtipo
  - tratamiento 206
- SUM, función
  - descripción básica 250
- SUM, funciones
  - descripción de formato
    - detallado 277
  - valores y argumentos 277
- SUMMARY, tabla
  - en sentencia CREATE
    - TABLE 843
- supergrupos 489
- supertipo
  - nombre de supertipo, convenios
    - para 77
- supervisor de sucesos
  - DROP, sentencia 1005
  - sentencia FLUSH EVENT
    - MONITOR 1054
  - SET EVENT MONITOR STATE,
    - sentencia 1166
- supervisores de sucesos
  - CREATE EVENT MONITOR,
    - sentencia 684
  - descripción 37
  - descripción del nombre 74
  - EVENT\_MON\_STATE,
    - función 334
- supresión de objetos SQL 1005
- T**
  - tabla
    - autorización para crear 835
    - creación de instrucciones de
      - sentencias de SQL 835



- tabla (*continuación*)
  - crear una tabla, otorgando autorización 1056
  - nombres en cláusula FROM 478
- tabla ADVISE\_INDEX
  - creación 1491
  - descripción detallada 1479
- tabla ADVISE\_WORKLOAD
  - descripción detallada 1482
- tabla base
  - definición 14
- tabla conectada por supresión 22
- tabla de objetos 147
- tabla resultante
  - definición 14
- tabla sujeto del desencadenante 37
- tabla unida 483
  - referencia de tabla 479
- tablas
  - actualización por fila y columna, sentencia UPDATE 1193
  - adición de columnas, ALTER TABLE 571
  - adición de comentarios al catálogo 628
  - autorreferente 20
  - base de datos de muestra 1427
  - base de datos relacional 9
  - base de datos relacional distribuida 43
  - calificación de un nombre de columna 144
  - cláusula FROM, convenios de denominación de subselección 478
  - clave de particionamiento 17
  - clave de unicidad 17
  - clave foránea 17
  - clave primaria 17
  - colocación 43
  - columnas generadas 565
  - con tipo, y desencadenantes 918
  - convenios de nombre 77
  - definición 14
  - dependiente 20
  - descendiente 20
  - descripción del nombre 77
  - designador para evitar ambigüedad 147
  - escalar, selección completa 149
  - esquemas 822
  - excepciones 1173, 1505
  - expresión de tabla anidada 149
  - expresión de tabla común 27
- tablas (*continuación*)
  - ID de autorización en nombre 79
  - índices 774
  - insertar filas 1082
  - mapa de particionamiento 42
  - mencionado en la sentencia ALTER TABLE 571
  - modificar definición 565
  - nombre de correlación 144
  - nombre en la sentencia LOCK TABLE 1092
  - nombres de correlación exclusivos
    - designadores de tabla 150
  - nombres en cláusula SELECT, diagrama de sintaxis 473
  - nombres expuestos en cláusula FROM 145
  - nombres no expuestos en cláusula FROM 145
  - otorgar privilegio de control 1072
  - otorgar privilegios 1070
  - padre 19, 20
  - renombrar, requisitos 1114
  - referencia de tabla 479
  - restricción del acceso, sentencia LOCK TABLE 1092
  - revocar privilegios 1131
  - seudónimo 670, 1008
  - subconsulta 149
  - suprimir utilizando la sentencia DROP 1005
  - tabla base 14
  - tabla resultante 14
  - temporal declarada 14
  - temporal global declarada 14
  - temporales en sentencia OPEN 1097
  - vistas de catálogo en tablas del sistema 1281
- tablas con tipo
  - convenios de nombre 78
  - descripción del nombre 78
- tablas de evaluación 232
- tablas de excepciones
  - estructura 1505
  - SET INTEGRITY, sentencia 1173
- tablas de resumen
  - REFRESH TABLE, sentencia 1110
- tablas de transición
  - en desencadenantes 38
- tablas resultante
  - resultado de consulta 471
- tablas resultantes intermedias 478, 486, 487, 495
- tablas temporales
  - OPEN, sentencia 1097
- tablas temporales declaradas
  - nombres de esquema en 77
- TABLE, cláusula
  - CREATE FUNCTION (Tabla externa), sentencia 723
  - DROP, sentencia 1017
  - referencia de tabla 479
  - sentencia COMMENT 636
- TABLE HIERARCHY, cláusula DROP, sentencia 1018
- TABLE\_NAME, función
  - descripción básica 250
  - descripción detallada 429
  - seudónimo 429
  - valores y argumentos 429
- TABLE\_SCHEMA, función
  - descripción básica 250
  - descripción detallada 431
  - seudónimo 431
  - valores y argumentos 431
- TABLESPACE, cláusula
  - sentencia COMMENT 636
- TAN, función
  - descripción básica 250
  - descripción detallada 434
  - valores y argumentos 434
- terminación
  - unidad de trabajo 640, 1139
- terminación de una unidad de trabajo 1139
- TIME, función
  - descripción básica 251
  - descripción detallada 435
  - valores y argumentos 435
- TIME, tipo de datos 853
- TIMESTAMP
  - función escalar WEEK 454
  - función escalar WEEK\_ISO 455
- TIMESTAMP, función
  - descripción básica 251
  - descripción detallada 436
  - valores y argumentos 436
- TIMESTAMP, tipo de datos 853
- TIMESTAMP\_ISO, función
  - descripción básica 251
  - descripción detallada 438
  - valores y argumentos 438
- TIMESTAMPDIFF, función
  - descripción básica 251

- TIMESTAMPDIFF, función
  - (*continuación*)
  - descripción detallada 439
  - valores y argumentos 439
- tipo de datos
  - abstracto 923
  - columnas del resultado 476
  - CREATE TYPE (Estructurado),
    - sentencia 923
  - estructurado 923
  - fila 923
- tipo de datos CHARACTER 851
- tipo de datos de hora 91
- tipo de datos de resultado
  - argumentos de COALESCE 120
  - cláusula VALUES de múltiples
    - filas 120
  - expresiones de resultado de
    - CASE 120
  - operador de conjunto 120
  - operandos 120
- tipo de datos definido por el usuario
  - nombre-tipo-diferenciado
    - CREATE TABLE,
      - sentencia 853
    - nombre-tipo-estructurado
      - CREATE TABLE,
        - sentencia 854
  - tipo de enlace de datos
    - descripción 95
  - tipo definido por el usuario (UDT)
    - descripción 96
  - tipo diferenciado
    - como operandos aritméticos 183
    - comparación 119
    - concatenación 180
    - constantes 131
    - descripción 74
    - DROP, sentencia 1005
  - tipos
    - descripción del nombre 77
  - tipos de datos
    - abstracto 603
    - ALTER TYPE, sentencia 603
    - BIGINT 90
    - columnas del resultado 477
    - compatibilidad entre
      - particiones 127
    - conversión entre 101
    - DATALINK 95
    - definido por el usuario
      - diferenciado 96
      - estructurado 96
      - referencia 96
    - definidos por el usuario 677
  - tipos de datos (*continuación*)
    - diferenciado 677
    - estructurado 603
    - fecha y hora 91
    - filas, modificar 603
    - función TYPE\_ID 445
    - función TYPE\_SCHEMA 447
    - promoción 100
    - serie de caracteres 87
    - tipo de referencia 99
    - tipo diferenciado 96
    - tipo estructurado 97
    - TYPE\_NAME, función 446
  - tipos de datos numéricos
    - resumen 90
  - tipos de referencia
    - comparaciones 120
    - conversión 102
    - descripción 99
    - función Deref 317
  - tipos definidos por el usuario (UDT)
    - adición de comentarios al
      - catálogo 628
    - conversión 102
    - tipos diferenciados
      - CREATE DISTINCT TYPE,
        - sentencia 677
    - tipos estructurados
      - sentencia CREATE
        - TRANSFORM, sintaxis 903
  - tipos diferenciados
    - CREATE DISTINCT TYPE,
      - sentencia 677
    - descripción 96
  - tipos estructurados
    - catálogo 1397
    - CREATE TRANSFORM, sentencia
      - sintaxis 903
    - descripción 97
    - DROP, sentencia 1005
    - invocación de métodos 204
    - tratamiento de los subtipos 206
    - variables del sistema
      - principal 158
  - TO, cláusula
    - GRANT, sentencia 1058, 1059,
      - 1062, 1065, 1074
  - transformaciones
    - DROP, sentencia 1005
  - TRANSLATE, función
    - descripción básica 252
    - descripción detallada 441
    - serie de caracteres 441
    - serie gráfica 441
    - valores y argumentos 441
  - tratamiento
    - subtipos 206
  - TRIGGER, cláusula de la sentencia
    - COMMENT 636
  - TRUNC o TRUNCATE, función
    - descripción básica 252
  - truncamiento
    - números 106
  - TYPE, cláusula
    - DROP, sentencia 1020
    - sentencia COMMENT 637
  - TYPE, predicado
    - formato 230
  - TYPE\_NAME, función
    - descripción básica 252
    - descripción detallada 446
    - valores y argumentos 446

## U

  - UCASE, función
    - descripción básica 252
    - descripción detallada 448
    - valores y argumentos 448
  - UNDER, cláusula
    - CREATE VIEW, sentencia 960
  - unicidad, claves
    - definición 17
  - unidad de trabajo
    - definición 28
    - destrucción de sentencias
      - preparadas 1108
    - referencia a sentencias
      - preparadas 1099
    - terminación destruye las
      - sentencias preparadas 1108
  - unidad de trabajo remota
    - descripción 45
  - unidades de trabajo
    - COMMIT, sentencia 640
    - crear nuevas 1139
    - en sentencia ROLLBACK 1139
    - iniciando cierres del cursor 1097
    - sentencia ROLLBACK,
      - efecto 1139
    - terminación 640
    - terminación sin guardar
      - cambios 1139
  - unión
    - consideraciones acerca de claves
      - de particionamiento 882
    - UNION, operador, papel en la
      - comparación
        - de selección completa 513

- unión de tablas
    - consideraciones acerca de claves de particionamiento 882
  - unión externa
    - tabla unida 479, 483
  - uniones
    - colocación de tablas 43
    - ejemplos 499
    - ejemplos de subselecciones 496
    - unión externa completa 484
    - unión externa derecha 484
    - unión externa izquierda 484
    - unión interna 484
  - UNIQUE, cláusula
    - ALTER TABLE, sentencia 579
    - CREATE INDEX, sentencia 775
    - CREATE TABLE, sentencia 873
  - UNIQUE, clave
    - CREATE TABLE, sentencia 860
  - unitario
    - signo más 181
    - signo menos 181
  - UPDATE, cláusula
    - sentencia GRANT (tabla, vista o apodo) 1073
    - sentencia REVOKE, eliminar privilegios 1133
  - UPDATE, sentencia
    - descripción detallada 1193
    - selección completa de fila 1198
  - UPPER, función
    - descripción detallada 448
    - valores y argumentos 448
  - USA
    - formato de indicación de fecha y hora 92
  - USER, registro especial 143
  - USING, cláusula
    - FETCH, sentencia 1051
    - OPEN, listado de variables del lenguaje principal en la sentencia 1094
    - sentencia EXECUTE 1036
  - USING DESCRIPTOR 1037
  - USING DESCRIPTOR, cláusula
    - OPEN, sentencia 1095
    - sentencia EXECUTE 1037
  - utilización de la hora en una expresión
    - TIME, función 435
- ## V
- valor
    - definición de datos 14
  - valor de columna compuesta 492
  - valor nulo, SQL
    - apariciones en filas duplicadas 473
    - asignación 106
    - columnas del resultado 475
    - condición desconocida 232
    - especificado por variable indicadora 153
    - expresiones-agrupación, utilizaciones permitidas 487
  - valor nulo en SQL
    - definición 84
  - valor por omisión
    - columna
      - ALTER TABLE, sentencia 576
      - CREATE TABLE, sentencia 861
  - valores de enteros pequeños de expresiones
    - SMALLINT, función 421
  - valores de longitud de bytes, lista para tipos de datos 359
  - valores de secuencia
    - generación 339
  - valores en SQL 83
  - valores enteros de expresiones
    - función INTEGER 353
  - VALUE, función
    - descripción básica 252
    - descripción detallada 449
    - valores y argumentos 449
  - VALUES, cláusula
    - INSERT, sentencia, carga de una fila 1084
    - número de valores, reglas 1084
    - selección completa 512
  - VALUES, sentencia
    - descripción detallada 1204
  - VALUES INTO, sentencia
    - descripción detallada 1205
  - VARCHAR
    - función escalar DOUBLE 329
    - función escalar WEEK 454
    - función escalar WEEK\_ISO 455
  - VARCHAR, función
    - descripción básica 253
    - descripción detallada 450
    - valores y argumentos 450
  - VARCHAR, tipo de datos 851
  - VARGRAPHIC, función
    - descripción básica 253
    - descripción detallada 452
    - valores y argumentos 452
  - variable del lenguaje principal
    - EXECUTE IMMEDIATE, sentencia 1042
  - variables de transición
    - en desencadenantes 38
  - variables del lenguaje principal de aplicación Assembler 1042
  - variables del sistema principal
    - aplicaciones REXX 615
    - asignación de valores de una fila 1145, 1205
    - BLOB 154
    - CLOB 154
    - DBCLOB 154
    - definición 151
    - descripción 75
    - diagrama de sintaxis 152
    - enlazar conjunto activo con cursor 1094
    - FETCH, sentencia 1051
    - identificador del lenguaje principal en 75
    - inserción en filas, sentencia INSERT 1084
    - PREPARE, sentencia 1100
    - reglas de declaración relacionadas con el cursor 978
    - sentencias de SQL incorporadas 539
    - sentencias de SQL incorporadas, declaración de comienzo 615
    - sentencias de SQL incorporadas, declaración de fin 1034
    - serie de sentencia, listado restringido
      - PREPARE, sentencia 1100
    - sustitución para marcadores de parámetros 1036
    - utilización incorporada, BEGIN DECLARE SECTION 615
  - variables indicadoras 153
  - variables indicadoras
    - descripción 153, 1042
    - variable del lenguaje principal, usos de declaración 153
  - variables SQL 642, 1233
  - VARIANCE, descripción detallada de la función 278
  - VARIANCE o VAR, función
    - descripción básica 253
  - vía de acceso de función 97
  - vía de acceso de SQL
    - registro especial CURRENT PATH 138
    - resolución 162

- vía de acceso de SQL (*continuación*)
    - tipos diferenciados 97
  - VIEW, cláusula
    - CREATE VIEW, sentencia 957
    - DROP, sentencia 1022
  - VIEW HIERARCHY, cláusula
    - DROP, sentencia 1022
  - vistas
    - actualizables 967
    - actualización de filas por
      - columnas, sentencia
      - UPDATE 1193
    - adición de comentarios al
      - catálogo 628
    - calificación de un nombre de
      - columna 144
    - cláusula FROM, convenios de
      - denominación de
      - subselección 478
    - clave foránea, restricciones de
      - referencia 15
    - creación 957
    - definición 15
    - esquemas 822
    - ID de autorización en
      - nombre 79
    - índice, relación con vista 16
    - insertable 967
    - insertar filas en tabla
      - visualizada 1082
    - no operativa 967
    - nombre de vista, convenios 78
    - nombres de columna 960
    - nombres en cláusula FROM 478
    - nombres en cláusula SELECT,
      - diagrama de sintaxis 473
    - nombres expuestos en cláusula
      - FROM 145
    - nombres no expuestos en
      - cláusula FROM 145
    - otorgar privilegios 1070
    - prevención de la pérdida de
      - definición de vista, WITH
      - CHECK OPTION 1199
    - privilegio CONTROL
      - límites en 1072
      - otorgar 1072
    - reglas al revocar privilegio 1134
    - revocar privilegios 1131
    - seudónimo 670, 1008
    - sólo lectura 967
    - suprimible 966
    - suprimir utilizando la sentencia
      - DROP 1005
  - vistas (*continuación*)
    - WITH CHECK OPTION, efecto
      - en UPDATE 1199
  - vistas actualizables 967
  - vistas con tipo
    - convenios de nombre 78
    - definir subvistas 960
    - descripción del nombre 78
  - vistas de catálogo
    - actualizables 1282
    - ATTRIBUTES 1287
    - BUFFERPOOLNODES 1289
    - BUFFERPOOLS 1290
    - CASTFUNCTIONS 1291
    - COLAUTH 1293
    - COLCHECKS 1294
    - COLDIST 1295
    - COLOPTIONS 1296
    - COLUMNS 1297
    - CONSTDEP 1303
    - CHECKS 1292
    - DATATYPES 1304
    - DBAUTH 1306
    - definición 32
    - EVENTMONITORS 1308
    - EVENTS 1310
    - FULLHIERARCHIES 1311
    - FUNCDEP 1312
    - FUNCMAPOPTIONS 1313
    - FUNCMAPPARMOPTIONS 1314
    - FUNCMAPPINGS 1315
    - FUNCPARMS 1316
    - FUNCTIONS 1318
    - HIERARCHIES 1324
    - INDEXAUTH 1325
    - INDEXCOLUSE 1326
    - INDEXDEP 1327
    - INDEXES 1328, 1400
    - INDEXEXPLOITRULES 1404
    - INDEXEXTENSIONDEP 1405
    - INDEXEXTENSIONMETHODS 1406
    - INDEXEXTENSIONPARMS 1407
    - INDEXEXTENSIONS 1408
    - INDEXOPTIONS 1332
    - KEYCOLUSE 1333
    - NAMEMAPPINGS 1334
    - NODEGROUPDEF 1335
    - NODEGROUPS 1336
    - PACKAGEAUTH 1337
    - PACKAGEDEP 1338
    - PACKAGES 1339
    - para tipos estructurados 1397
    - PARTITIONMAPS 1344
    - PASSTHROUGHAUTH 1345
    - PREDICATESPECS 1409
  - vistas de catálogo (*continuación*)
    - PROCEDURES 1346
    - PROCOPTIONS 1349
    - PROCPARMOPTIONS 1350
    - PROCPARMS 1351
    - REFERENCES 1353
    - REVTYPEMAPPINGS 1354
    - SCHEMAAUTH 1356
    - SCHEMATA 1357
    - SEQUENCES 1358
    - SERVEROPTIONS 1360
    - SERVERS 1361
    - sólo lectura 1282
    - STATEMENTS 1362
    - SYSDUMMY1 1286
    - SYSSTAT.COLDIST 1385
    - SYSSTAT.COLUMNS 1386
    - SYSSTAT.FUNCTIONS 1388
    - SYSSTAT.INDEXES 1390
    - SYSSTAT.TABLES 1394
    - TABAUTH 1363
    - TABCONST 1365
    - TABLES 1366
    - TABLESPACES 1371
    - TABOPTIONS 1373
    - TBSPACEAUTH 1374
    - TRANSFORMS 1410
    - TRIGDEP 1375
    - TRIGGERS 1376
    - TYPEMAPPINGS 1378
    - USEROPTIONS 1380
    - VIEWDEP 1381
    - IEWS 1382
    - visión general 1281
    - WRAPOPTIONS 1383
    - WRAPPERS 1384
  - vistas de sólo lectura 967
  - vistas insertables 967
  - vistas no operativas 967
  - vistas suprimibles 966
  - visualizar
    - información en línea 1644
- ## W
- WEEK, función
    - descripción básica 253
    - descripción detallada 454
    - valores y argumentos 454
  - WEEK\_ISO, función
    - descripción básica 253
    - descripción detallada 455
    - valores y argumentos 455
  - WHENEVER, sentencia
    - cambio del flujo de control 539
    - descripción detallada 1207

- WHERE, cláusula
  - DELETE, sentencia 993
  - función de búsqueda, subselección 486
  - sentencia UPDATE, búsqueda condicional 1198
- WHERE CURRENT OF, cláusula
  - DELETE, sentencia, utilización de DECLARE CURSOR 994
  - UPDATE, sentencia 1198
- WHILE, sentencia 1251
- WITH, cláusula
  - CREATE VIEW, sentencia 963
  - INSERT, sentencia 1084
- WITH, expresión de tabla común 518
- WITH CHECK OPTION, cláusula
  - CREATE VIEW, sentencia 964
- WITH DEFAULT, cláusula
  - ALTER TABLE, sentencia 574
- WITH GRANT OPTION, cláusula
  - GRANT, sentencia 1075
- WITH HOLD, cláusula
  - DECLARE CURSOR, sentencia 977
- WITH OPTIONS, cláusula
  - CREATE VIEW, sentencia 961
- WORK, palabra clave
  - COMMIT, sentencia 640

## Y

- YEAR, función
  - descripción básica 253
  - descripción detallada 456
  - valores y argumentos 456



---

## Cómo ponerse en contacto con IBM

Si tiene un problema técnico, repase y lleve a cabo las acciones que se sugieren en la *Guía de resolución de problemas* antes de ponerse en contacto con el Centro de Asistencia al Cliente de DB2. Dicha guía sugiere información que puede reunir para ayudar al Centro de Asistencia a proporcionarle un mejor servicio.

Para obtener información o para solicitar cualquiera de los productos de DB2 Universal Database, consulte a un representante de IBM de una sucursal local o a un concesionario autorizado de IBM.

Si vive en los Estados Unidos, puede llamar a uno de los números siguientes:

- 1-800-237-5511 para obtener soporte técnico
- 1-888-426-4343 para obtener información sobre las opciones de servicio técnico disponibles

---

### Información sobre productos

Si vive en los Estados Unidos, puede llamar a uno de los números siguientes:

- 1-800-IBM-CALL (1-800-426-2255) o 1-800-3IBM-OS2 (1-800-342-6672) para solicitar productos u obtener información general.
- 1-800-879-2755 para solicitar publicaciones.

**<http://www.ibm.com/software/data/>**

Las páginas World Wide Web de DB2 ofrecen información actual sobre DB2 referente a novedades, descripciones de productos, planes de formación, etc.

**<http://www.ibm.com/software/data/db2/library/>**

La biblioteca técnica de servicio y de productos DB2 ofrece acceso a preguntas frecuentemente formuladas (FAQ), arreglos de programa, manuales e información técnica actualizada sobre DB2.

**Nota:** Puede que esta información sólo esté disponible en inglés.

**<http://www.elink.ibm.com/pbl/pbl/>**

El sitio Web para el pedido de publicaciones internacionales proporciona información sobre cómo hacer pedidos de manuales.

**<http://www.ibm.com/education/certify/>**

El Programa de homologación profesional contenido en el sitio Web de IBM proporciona información de prueba de homologación para diversos productos de IBM, incluido DB2.

**ftp.software.ibm.com**

Conéctese como anónimo (anonymous). En el directorio /ps/products/db2 encontrará programas de demostración, arreglos de programa, información y herramientas referentes a DB2 y a muchos otros productos.

**comp.databases.ibm-db2, bit.listserv.db2-l**

En estos foros de discusión de Internet los usuarios pueden explicar sus experiencias con los productos DB2.

**En Compuserve: GO IBMDB2**

Entre este mandato para acceder a los foros referentes a la familia de productos DB2. Todos los productos DB2 tienen soporte a través de estos foros.

Para conocer cómo ponerse en contacto con IBM desde fuera de los Estados Unidos, consulte el Apéndice A del manual *IBM Software Support Handbook*. Para acceder a este documento, vaya a la página Web siguiente: <http://www.ibm.com/support/> y luego seleccione el enlace "IBM Software Support Handbook", cerca del final de la página.

**Nota:** En algunos países, los distribuidores autorizados de IBM deben ponerse en contacto con su organización de soporte en lugar de acudir al Centro de Asistencia de IBM.







SC10-3497-01

